НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Клієнт-серверна система підтримки учбового процесу»

Виконав:
студент IV курсу, групи ІО-63
Божок Роман Юрійович _____

Керівник:
старший викладач
Алещенко Олексій Вадимович _____

Консультант нормоконтроль:
проф. д. т. н.
Сімоненко Валерій Павлович _____

Рецензент:

Радченко _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ СЕРГІЙ СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Божку Роману Юрійовичу

1. Тема проєкту «Клієнт-серверна система підтримки учбового процесу», керівник проєкту Алещенко Олексій Вадимович, старший викладач, затверджені наказом по університету від «07» травня 2020 р. №1081-С

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту *Технічна документація. Клієнт-серверна система. База даних MySQL. Середовище розробки IntelliJ IDEA, Java.*

4. Зміст пояснювальної записки *Аналіз предметної області, дослідження методики побудови клієнт-серверних систем на принципі MVC, розробка системи підтримки учбового процесу.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) функціональна схема, схема взаємодії, структурна схема.

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	*Затвердження теми роботи*	*01.09.2019*	
2	*Вивчення та аналіз завдання*	*30.03.2020*	
3	*Розробка архітектури та загальної структури систем*	*10.04.2020*	
4	*Розробка структур окремих підсистем*	*22.04.2020*	
5	*Програмна реалізація системи*	*04.05.2020*	
6	*Оформлення пояснювальної записки*	*17.05.2020*	
7	*Передзахист*	*26.05.2020*	
8	*Захист*	*16.06.2020*	

Студент Роман БОЖОК

Керівник Олексій АЛЕЩЕНКО

Анотація

Дана бакалаврська робота присвячена розробці клієнт-серверної системи, за допомогою якої може здійснюватися підтримка учбового процесу. Ця система має підсистему у вигляді тесту. Система не має реєстрації, що полегшує роботу для людей з низьким рівнем комп'ютерної грамотності. Окрім того результати отримані у ході тестування зберігаються для ведення статистики.

Annotation

This bachelor's thesis is devoted to the development of a client-server system that can support the educational process. This system has a subsystem in the form of a test. The system is unregistered, which makes it easier for people with low computer literacy. In addition, the results obtained during testing are stored for statistical purposes.

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня__бакалавр___

на тему: “Клієнт-серверна система підтримки учбового процесу”

Київ – 2020 року

Технічне завдання до *дипломної* роботи

Зміст

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Клієнт-серверна система підтримки учбового процесу». Область застосування системи: альтернатива існуючим системам підтримки учбового процесу у навчальних закладах.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки цієї системи є завдання на виконання дипломного проєкту, підтримки учбового процесу, затвердженого кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою мого проєкту є розробка системи, що має проводити тестування, та у режимі реального часу надсилати отримані дані через систему в базу даних.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки мого проєкту служать науково-технічна література, довідники по шаблонам проєктуванням, публікації в спеціалізованих виданнях, та публікації в мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

- Проводити тестування за допомогою браузера;
- Отримувати результати одразу після проходження тесту;
- Можливість покращити систему додавши новий функціонал.

5.2. Вимоги до програмного забезпечення

- Операційна система Linux, macOS, Windows;
- Доступ до мережі Інтернет;
- Браузер Google Chrome, Firefox, Safari або Microsoft Edge.

5.3. Вимоги до апаратної частини

- Джерело живлення 220 В для локального комп’ютера;
- Доступ до мережі Інтернет;
- Локальний або хмарний комп’ютер.

6. **ЕТАПИ РОЗРОБКИ**

	Дата
Вивчення літератури	30.01.2020
Складання і узгодження технічного завдання	11.02.2020
Створення модулів системи, що розробляється	14.03.2020
Тестування окремих модулів системи	10.04.2020
Доопрацювання, налагодження і виправлення помилок	04.05.2020
Оформлення документації дипломної роботи	17.05.2020

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

до дипломної роботи
освітньо-кваліфікаційного рівня__бакалавр___

на тему: “ Клієнт-серверна система підтримки учбового процесу ”

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП.467100.001 ТЗ	Технічне завдання	4	
3	А4	ДП.467100.002 ВП	Відомість проєкту	1	
4	А4	ДП.467100.003 ПЗ	Пояснювальна записка	56	
5	А3	ДП.467100.004 Д1	Схема структурна Діаграма класів	1	
6	А3	ДП.467100.005 Д2	Схема функціональна Блок-схема алгоритму	1	
7	А3	ДП.467100.006 Д3	Схема взаємодії	1	

					ІАЛЦ.467100.002 ВП		
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Божок Р.Ю.*			*Відомість дипломного проєкту*	*Літ.*	*Арк.* 2 / *Акрушів* 1
Перевір.		*Алещенко О.В.*					
Н. Контр.		*Сімоненко В.П.*				*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-63*	
Затверд.							

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

на тему: “Клієнт-серверна система підтримки учбового процесу”

Київ – 2020 року

ЗМІСТ

ВСТУП

Головним завданням даної роботи є створення системи підтримки учбового процессу, а, зокрема, тестування. Дана система може бути застосована на практиці в якості тестуваня студентів або школярів на дистанційному навчанні. Ця тема стала актуальною під час карантину, так як усі вищі та середні навчальні заклади перейшли на дистанційне навчання.

У вищих навчальних закладах ці системи почали впроваджувати ще давно, тому дистанційне навчання для них не стало великою проблемою. А ось у закладах середньої освіти, зокрема школах, таких систем не має. І під час карантину, вчителі зіткнулися з проблемою, як швидко та зручно підтримати учбовий процес для дітей. Є аналоги, таких систем, але вони або частково комерційні, або мають не інтуїтивний інтерфейс та складну реєстрацію. В школах немає спеціальних курсів, де дітей та вчителів навчають як користуватися такими системами, тому постає питання в написанні системи, яка буде зручною як для вчителів так і для школярів.

Клієнт-серверна архітектура – це шаблон проєктування програмного забезпечення та концепція, яка домінує серед веб-застосунків і налаштовує взаємодію та пересилку даних.

Сервери не мають залежностей від інших серверів, паралельно також працють клієнти. Клієнт із серверами не має жорстких звязків, деякі сервери можуть оброблювати декілька різних клієнтів, з іншого боку, клієнт має можливість посилати запити на різні сервери.

Згідно з усіма наведеними вище твердженнями, поставленанаступна задача: створити систему підтримки учбового процессу у вигляді тесту. Відсутність реєстрації для учнів робить цю систему зручною в користуванні. Відсутність реєстрації має на увазі те що в системі є логін та пароль які адміністратор відсилає користувачам. Додатковими, супутніми завданнями були статистика по класах та школах.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ ГРАФІЧНИХ РЕДАКТОРІВ

1.1. Опис завдання

Завдання даної роботи полягає в створенні клієнт-серверної системи, яка дозволить користувачам (як учням, так і вчителям виконувати тестування швидко та зручно. Система повинна полегшити дистанційне тестування.

Система підтримки учбового процесу, яка виконується в даній бакалаврської роботи, включає в себе підсистему, що дозволяє тестування студентів або школярів. Такою підсистемою цілком може бути клієнт-серверна система.

Клієнт-серверна система дозволить користувачеві проходити тести. Додаткова функція системи є такі особливості як ведення обліку оцінок студентів або школярів.

1.2. Аналіз існуючих рішень

Існує багато систем підтримки учбового процесу. Одним з найвідоміших є Google Classroom.

Менш широко використовується система під назвою TeacherDashboard for Microsoft Office 365. Він вже більше спеціалізований для побудови тестів, ніж Classroom і не потребує засобів для тестування таких як Google форми або Quizlet. Отже, він не такий ресурсномісткий і в деяких аспектах складний у використанні як ClassRoom.

Крім усього іншого варто зауважити, що TeacherDashboard є комерційним і для вільного і необмеженого його використання потрібне придбання ліцензії, а для використання Google Classroom потрібен зареєстрований гугл акакунт для всіх користувачів, для студентів це не проблема, а для школярів може бути проблемою.

Для використання локально розроблений пакет програм під назвою UniTest System, пропонує основні можливості по розробці компютерних

тестів, проведення тестування та детального аналізу результатів. Програма пропонує також додавання до тестів документів, відео файлів, вбудований редактор, подібний MS Word, який дозволяє створювати красиві тести. Все перераховане вище надає користувачеві широкі можливості, але цей програмний продукт був розробленний у 2009 році, тому зараз є не дуже актуальним, зокрема в школах в яких немає інтернету. Ця програма поширюється вільно, завантажується з офіційного сайту.

Тепер розглянемо функціонал та складність творення тестів, наприклад в Classroom.

1.2.1. Google classroom [5]

Для використання classroom потрібно мати Google акаунт зображено на рисунку. 1.1.

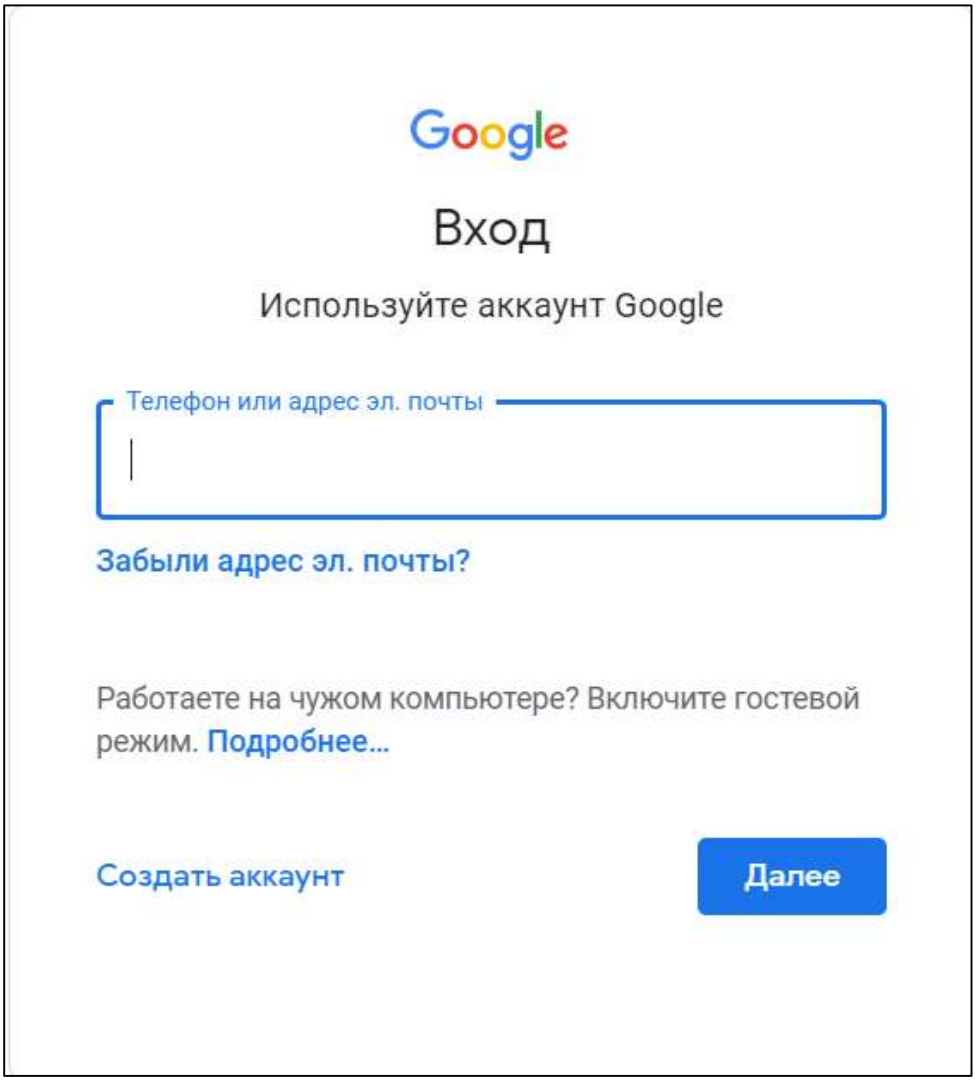


Рис. 1.1. Логін форма в Google [5]

Google акаунт - спеціалізована платформа, вона надає той набір інструментів, який необхідний для створення облікового запису в інтернеті. Набір додаткових опцій дозволяє використовуваті ресурси google без додаткових регістрацій, але сама регістрація має вікові обмеження, тому учні молодших класів не зможуть використовувати цю платформу.

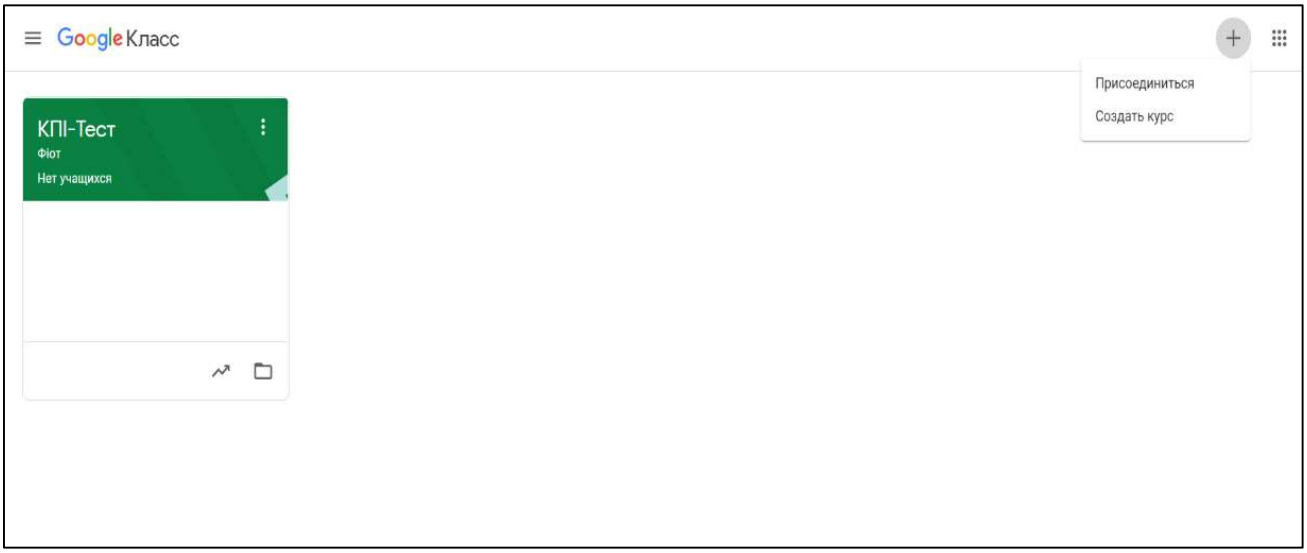


Рис. 1.2. Наступна форма [5]

Після успішної авторизації нас вітає стандатрний інтерфейс (Рис. 1.2), у якому є відкриті курси та вибір, створити новий курс або приєднатися до створенного.

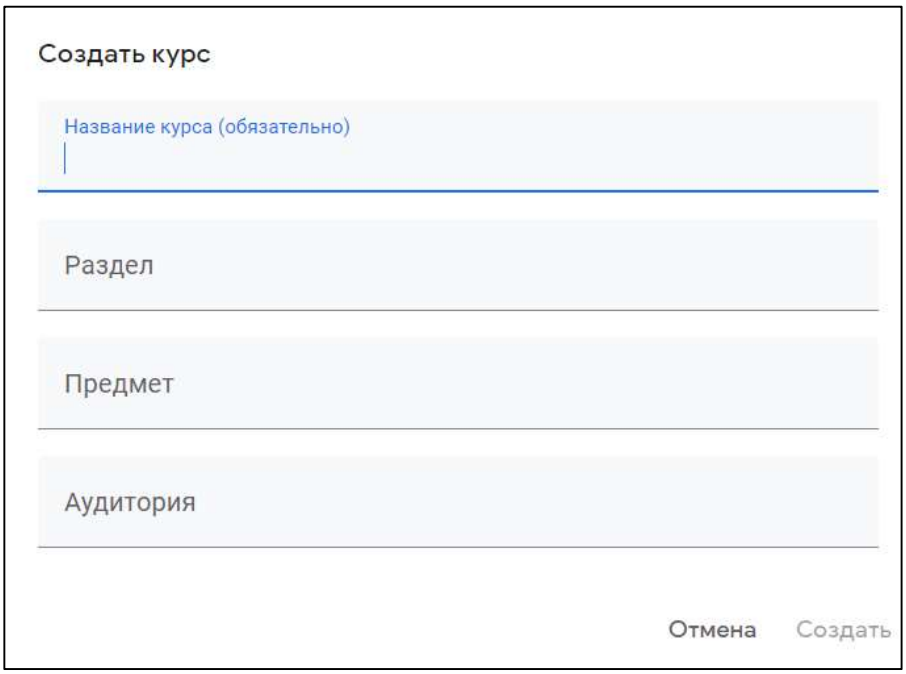


Рис. 1.3. Створення курсу [5]

Створення курсу полягає у тому, що би заповнити наступні поля як показано на рисунку 1.3.

Для того щоб потрапити на цей курс, google classroom формує код, за допопомогою якого, будь який студент може під’єднатися до курсу(Рис.1.4).

Рис. 1.4. Код курсу [5]

Для створення тесту, в цій платформі, потрібно вікористовувати інші ресурси, такі як Google форми або Quizlet.

Далі ми прикріплюємо ці форми до курсу, підписуємо, та натискаємо створити, завдання створенно.

Завдання відобразится у всіх кто під’єднався до курсу і його виконання буде можливим.

Плюси:

- простий інтерфейс,
- не складна система підключення авторизованих студентів,
- синхронізованість критеріїв оцінювання,
- підтримка на мобільних пристроях.

Мінуси:

- складна регістрація з обмеженням по віку,

- якщо використовувати іншу систему тестування, вчителю потрібно виставляти оцінку самому,
- не пристосована для локального використування.

1.2.2. Google форми [6]

Головне вікно форми зображено на рисунку 1.5.

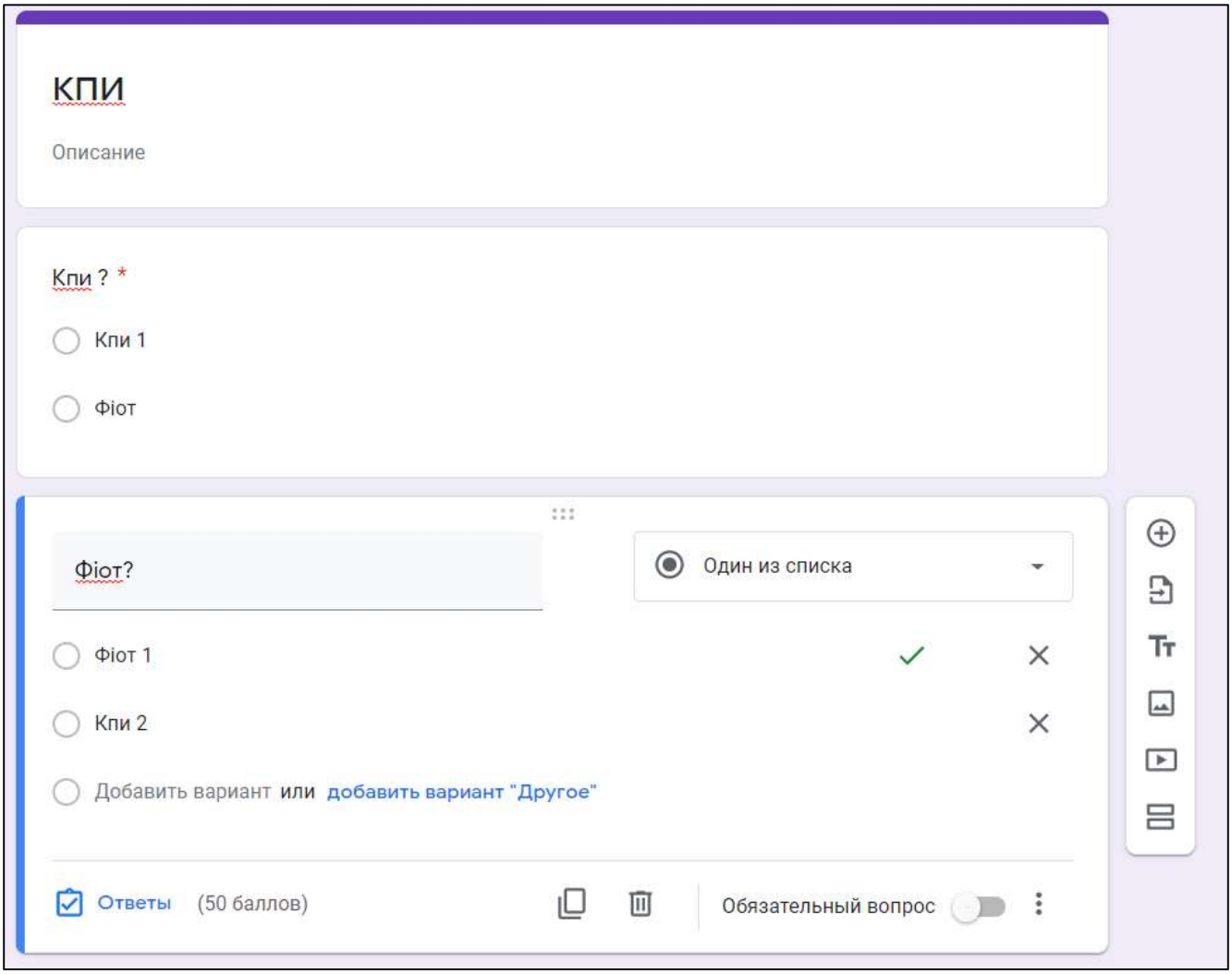


Рис. 1.5. Головне вікно для створення Google форми [6]

Принцип роботи форми - робота з питаннями та відповідями. Є повний функціонал для створення тесту. Також є можливість доповнити тест іншими питаннями із іншої форми, додати зображення, відео чи розділ. Встановлення балу за відповідь, позначати обов'язкові відповіді, але цей функціонал більш зручний для опитування. Багатий функціонал відповідей, який показаний на рисунку 1.6.

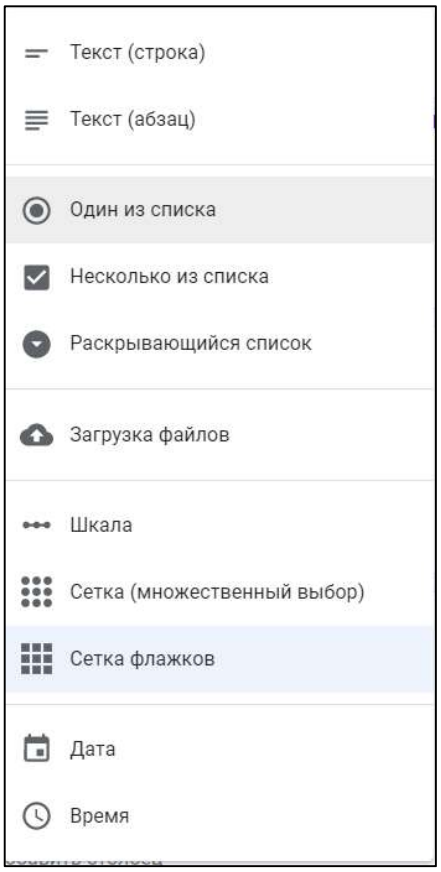


Рис. 1.6. Варіанти відповіді [6]

Отже, форми пристосованні для тестування та опитувань, але для зображення візуалізованих тестів ні.

1.2.3. Веб-застосунок Quizlet [7]

Головне вікно застосунку зображено на рисинку 1.7

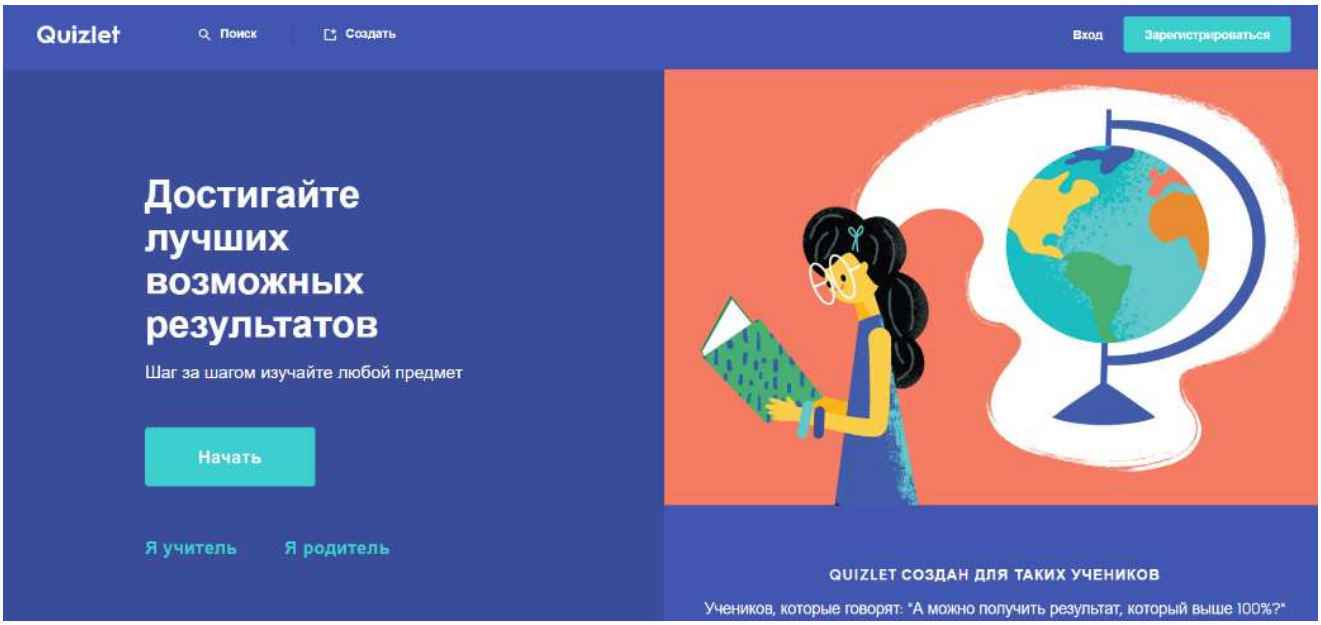


Рис. 1.7. Головне вікно програми Diagram Designer [7]

Quizlet – це веб-застосунок дуже схожий на google форми та google classroom. З більш зручним інтрефейсом, може використовуватися в google classroom(рис. 1.9), але без синхронізації оцінок. Quizlet безкоштовний, але буває і підписка Plus за $ 10 - вона дозволить завантажувати власні зображення і створювати необмежену кількість курсів . Має підтримку на мобільних пристроях. Підримка авторизації через гугл акакунт .

Має аналогічну систему додавання учнів у курс, але через посилання.(Рис.1.8)

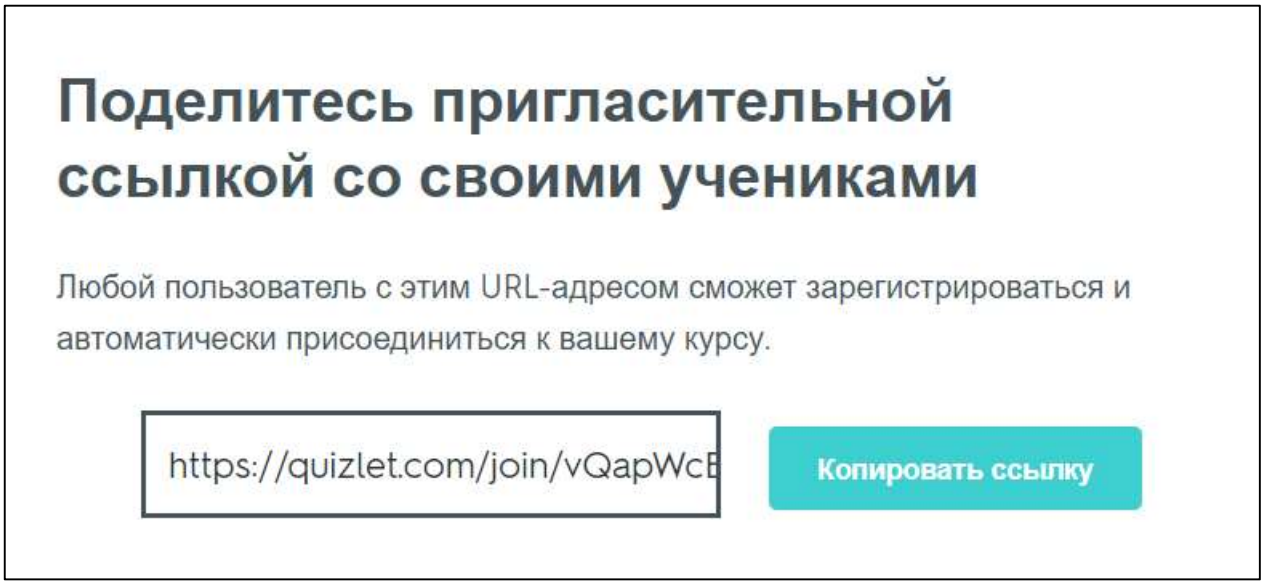


Рис. 1.8. Посилання на курс [7]

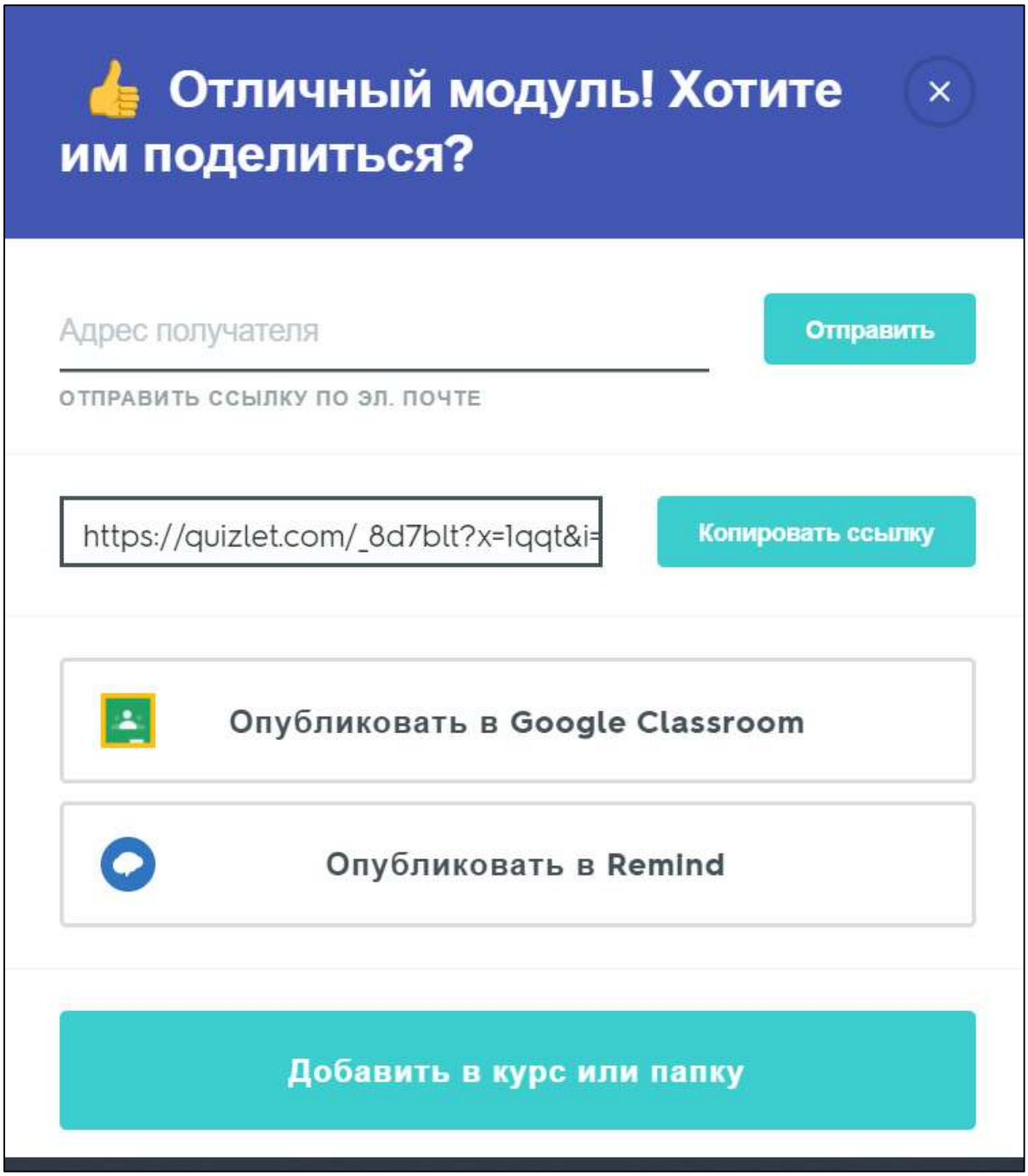


Рис. 1.9. Готовий модуль [7]

1.2.4. Програма для створення тестів UniTest System

Головне вікно програми зображено на рисунку 1.10

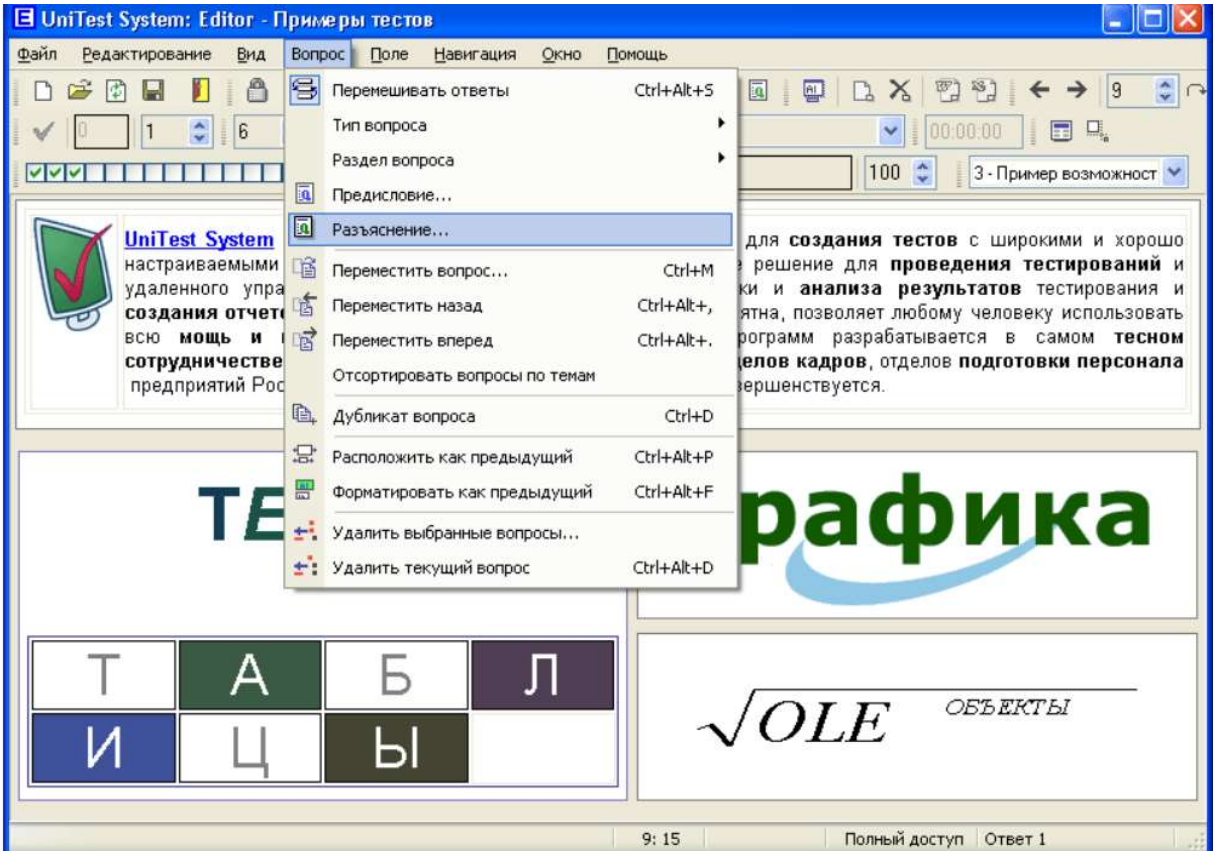


Рис 1.10. Головне вікно програми UniTest System

Дана програма дозволяє створювати тести локально. В режимі редагування може відображати питання тесту так, як вони будуть виглядати при тестуванні. Вставка даних досить зручна з інших програм. Програма має “горячі клавіші”. Також є обмежений доступ та режим блокування, який допомагає забезпечити безпеку данних. Режим розмітки допомагає простими претягуваннями і обробкою, розробити зрозумілий тест. Режим тест застосовується для проведення тестування .

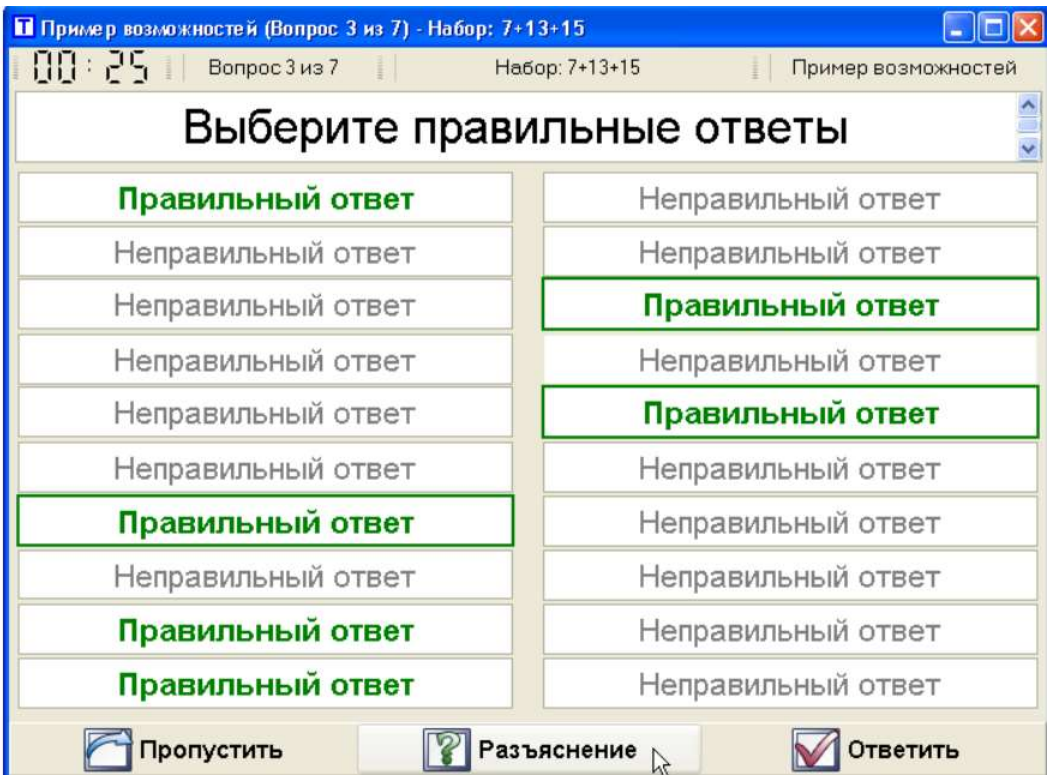


Рис. 1.11. Вид тесту

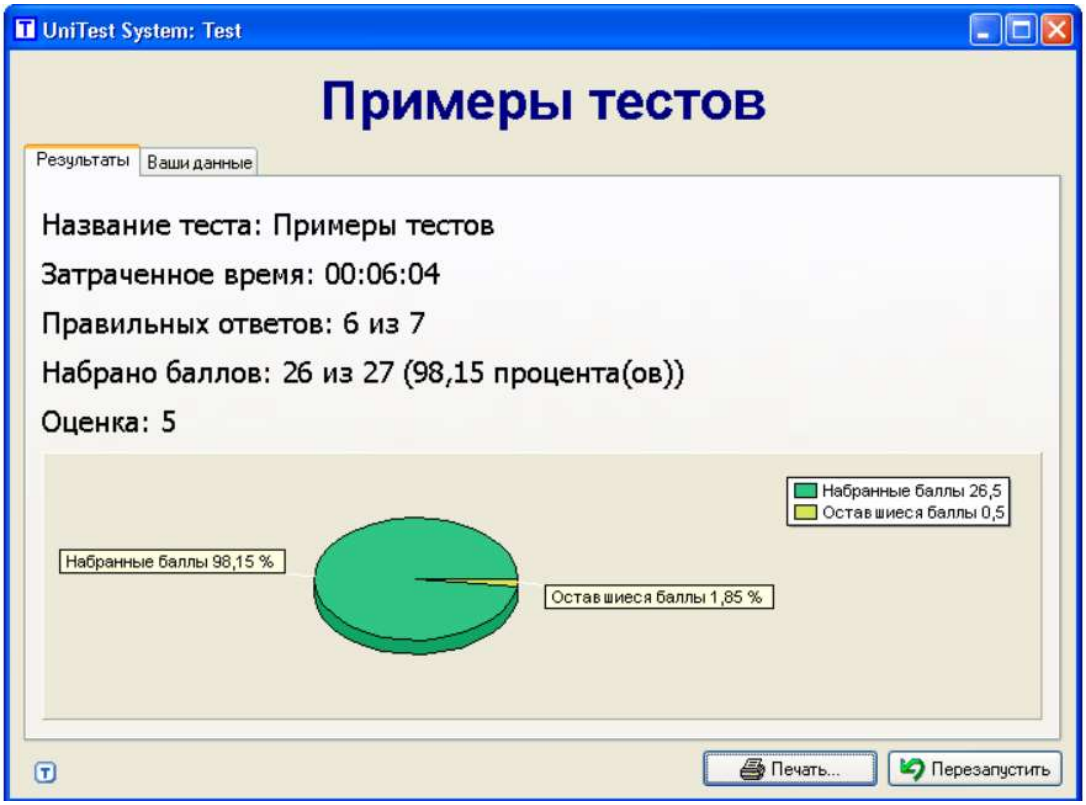


Рис. 1.12. Результат тесту

Тестування може відбуватися в віконному або повноекранному режимі. Всі поля тесту забезпечені вспливаючими підсказками, можливе

відклання питання. Всі результати тестування записуються в базу данних . Результат виводятся в балах з оцінкою(Рис.1.12).

Плюси :

- Зручний інтерфейс
- Локальне використування
- Є захист данних
- Тести займають мало пам’яті на машині
- Підтримка різних мов
- Адміністрація користувачів
- Підтримка багатьох типив питаннь
- Маштабування елементів тесту

Мінуси:

- Программа 2009 року
- Допотопний інтерфейс
- Результат тестування в одній системі оцінювання
- Комерційний
- Не має підтримки мобільних пристроїв

1.2.5. Веб-застосунок LearningApps.org [8]

Головне вікно програми зображено на рисунку 1.13.

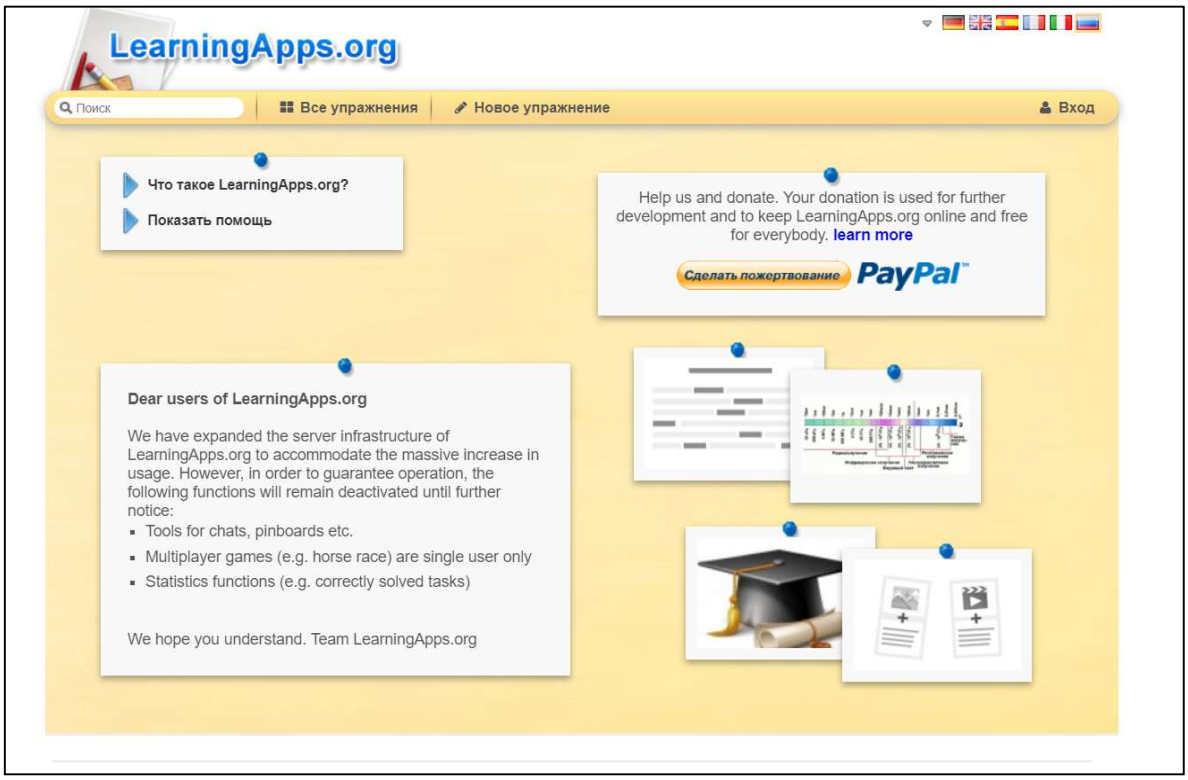


Рис 1.13. Головне вікно LearningApps.org [8]

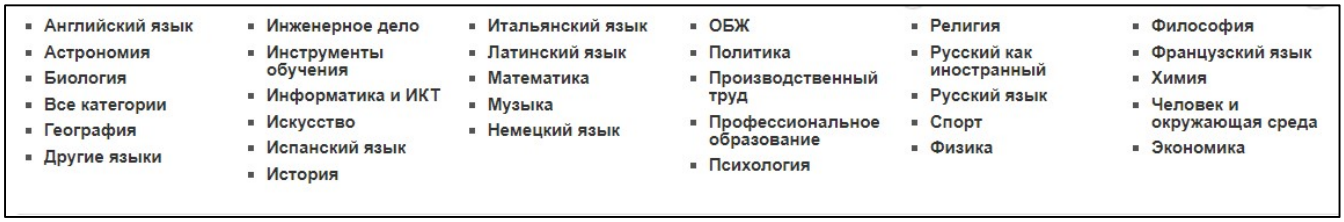


Рис 1.14. Вибір предметів [8]

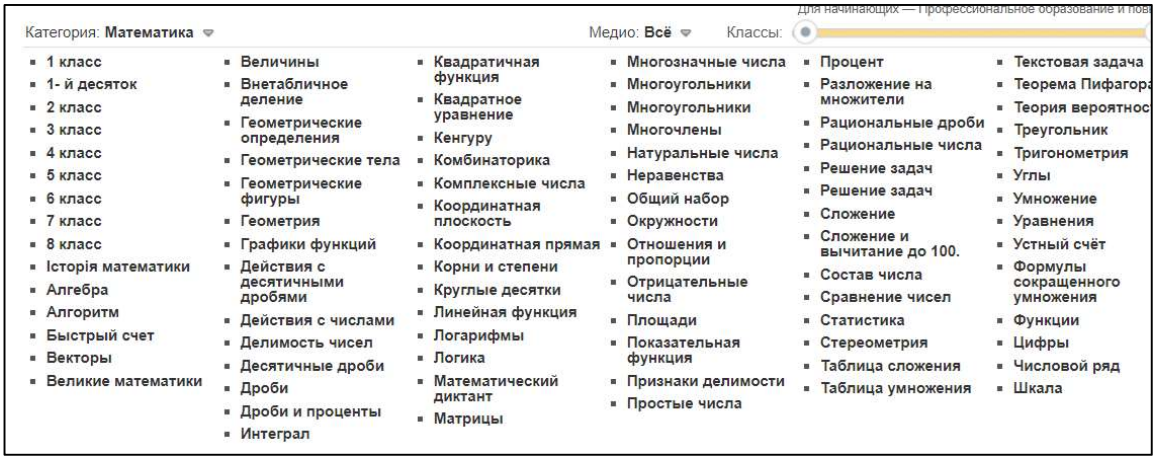


Рис 1.15. Вибір підтем [8]

LearningApps.org - веб-застосунок для підтримки навчання та процессу викладання за допомогою інтерактивних модулів. Має велику кільтість модулів по різним предметам(Рис.1.14). Кожний предмет включає в собі багато підтем як показано на рисунку 1.15. Привабливі тести(рис.1.16,1.17), які можливо розробити самому, велика база тестів.

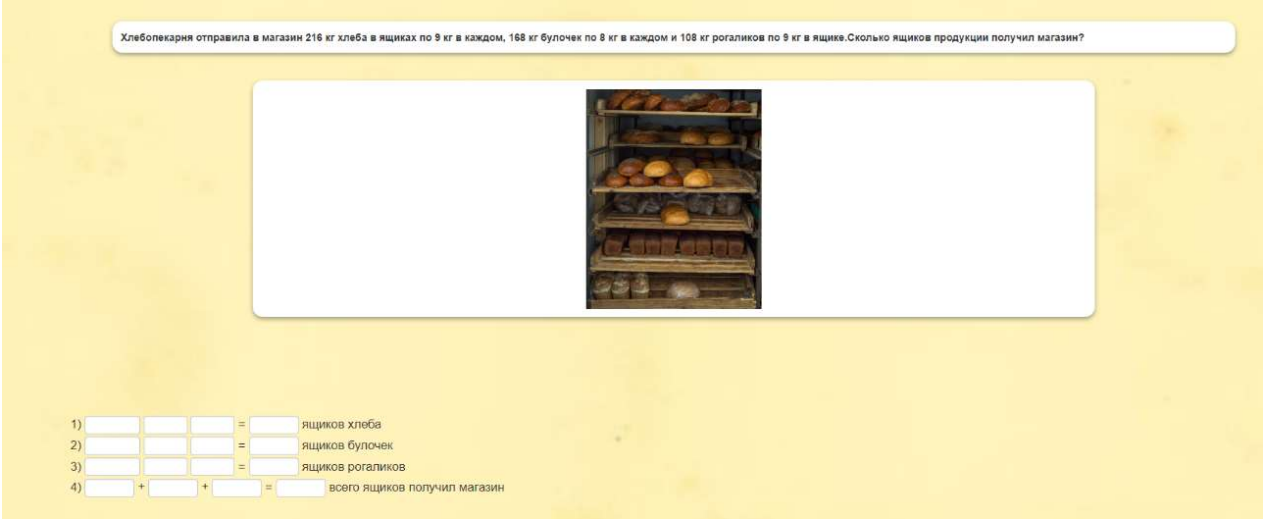


Рис 1.16. Приклад тесту [8]

Рис 1.17. Приклад тесту “Змагання” [8]

Недоліки:

Досить складний інтерфейс, який сопроводжується підказками. Не має оцінювання, кабінету для створення курсу, тільки створення модулів.

1.2.6. KPI Tests - система онлайн тестування студентів.

Головне вікно програми зображено на рисунку 1.18.

Рис 1.18. Головне вікно KPI Tests

KPI Tests – розробка студентів, які вирішили що сучасний рівень дистанційного навчання не гідний ні студен, ні викладач. Ця система дозволяє моментально і без особливих зусиль виконувати завдання викладачів.

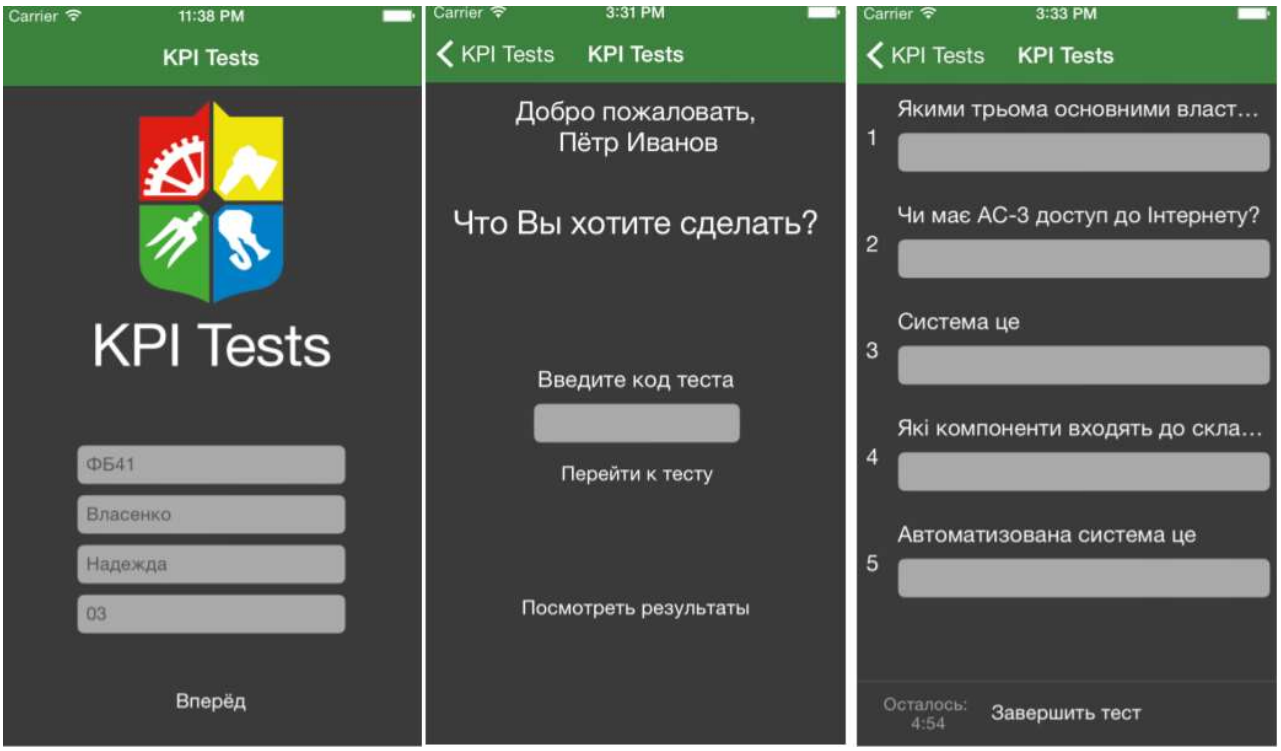


Рис 1.19. Функціонал KPI Tests

Плюси:

- зручний інтерфейс
- спеціальна розробка для КПІ
- кросплатформність

Мінуси:

- пристосованна лише для КПІ
- розробка студентів
- не готовий продукт
- не має підтримки у браузері

1.2.7. Проєкт «Всеосвіта»[9]

Головне вікно програми зображено на рисунку 1.20.

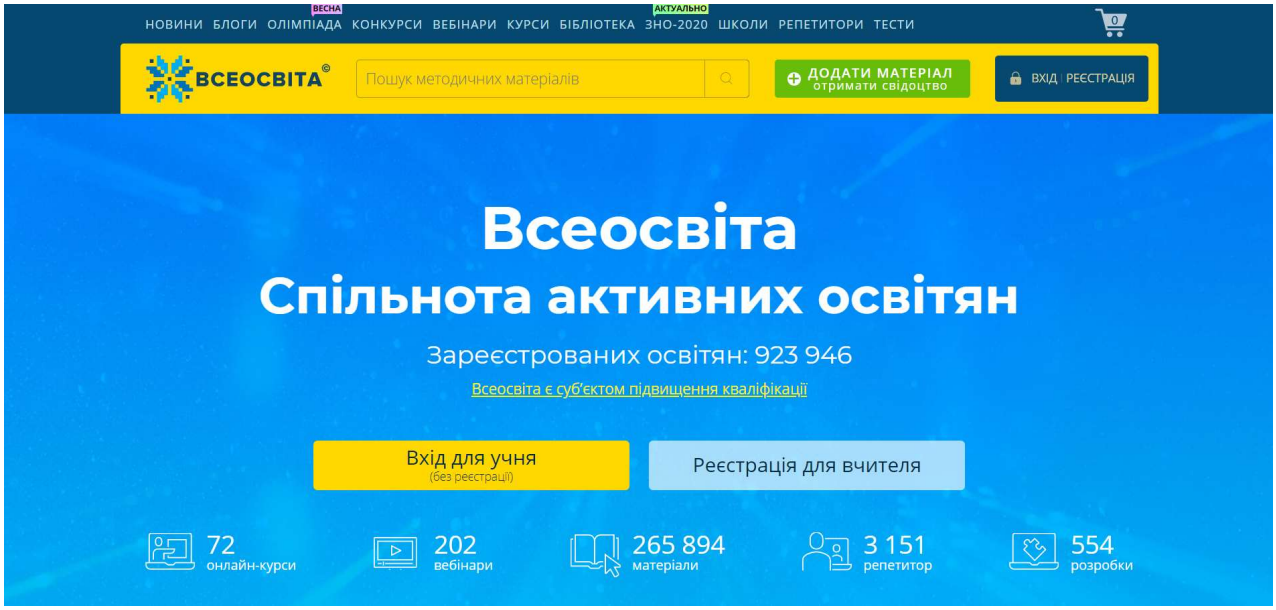


Рис 1.20. Головне вікно проєкту «Всеосвіта» [9]

Проєкт «Всеосвіта» - це новий та переспективний освітній ІТ-проєкт, який подає надії на цифрову освіту. Як показанно на рисунку 1.20, в цьому проєкті є 72 онлайн-курсів, 202 вебінара, 265 894 різних матеріалів та 3 151 репетиторів, які допоможуть вам розібратися за матеріалом, звістно не безкоштовно .

Виконавши не складну регістрацію (рис 1.21) та підтвердивши поштову скриню, відкривається всі можливості цього проєкту, як для вчителя.

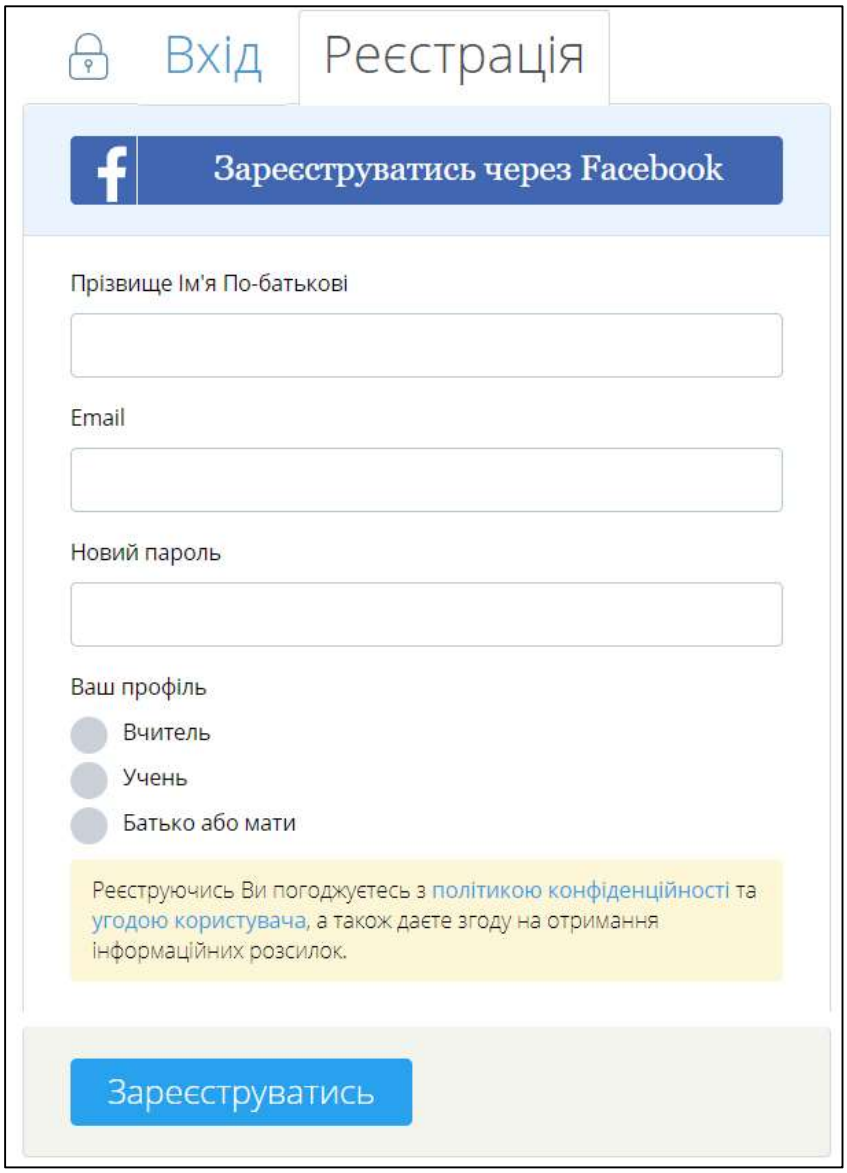


Рис 1.21. Регістраційна форма проєкту «Всеосвіта» [9]

Створити тест для учнів можливе, з використанням інших, або написати його власноруч .

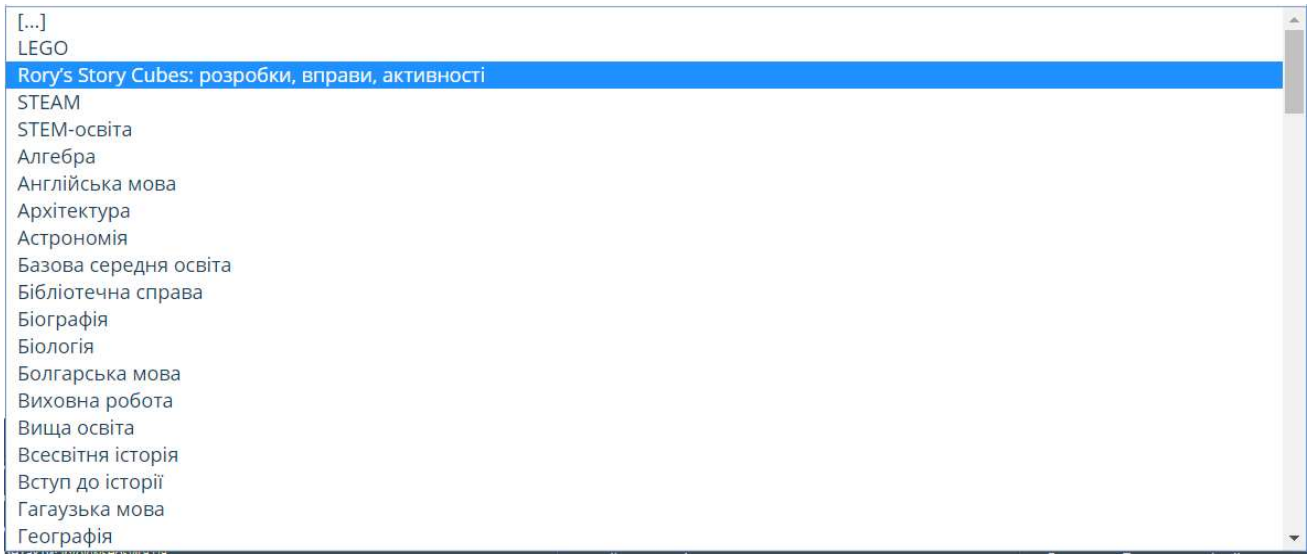


Рис 1.22. Тематика тестів [9]

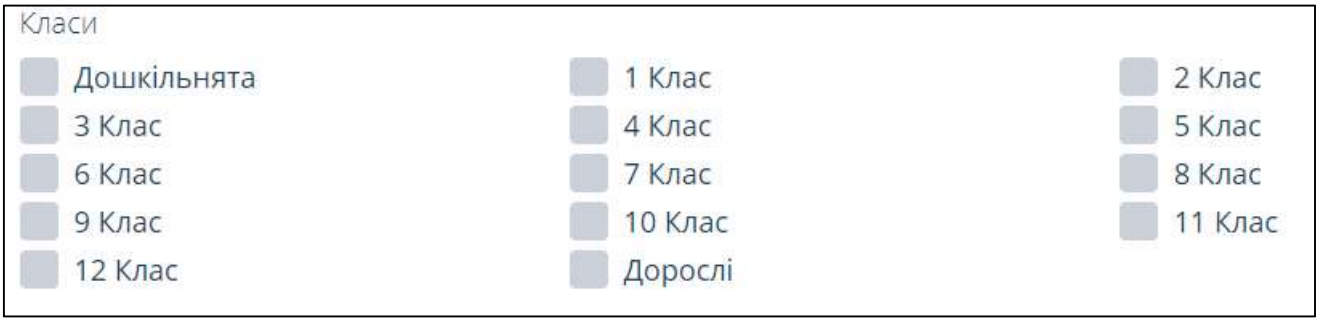


Рис 1.23. Вибір вікового орієнтиру [9]

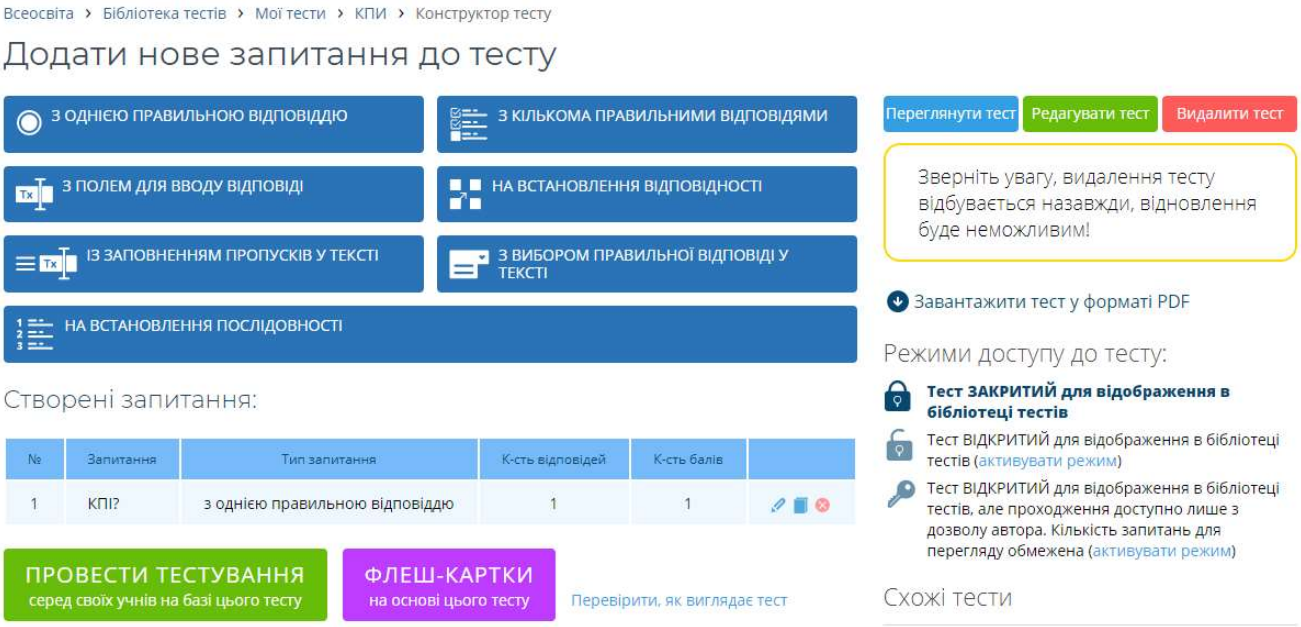


Рис 1.24. Типи питань [9]

Тематика тестів дуже різноманітна (рис 1.22) від шкільних предметів до курсів цифровоії освіти, або гуртків таких як LEGO. Також свій тест

можливо спеціалізувати віковим орієнтиром(рис 1.23). Віковими орієнтири є класи в школі, дошкільнята та дорослі. Тестова система має підтримку багатьох типів питань (рис 1.24):

- з однією правильною відповіддю,
- з кількома правильними відповідями,
- на встановлення послідовності,
- із заповненням пропусків у тексті,
- з вибором паравильнох відповіді у тексті.

Тест можливо зберегти у бібліотеку та надати доступ для використання іншим користувачам власноруч .

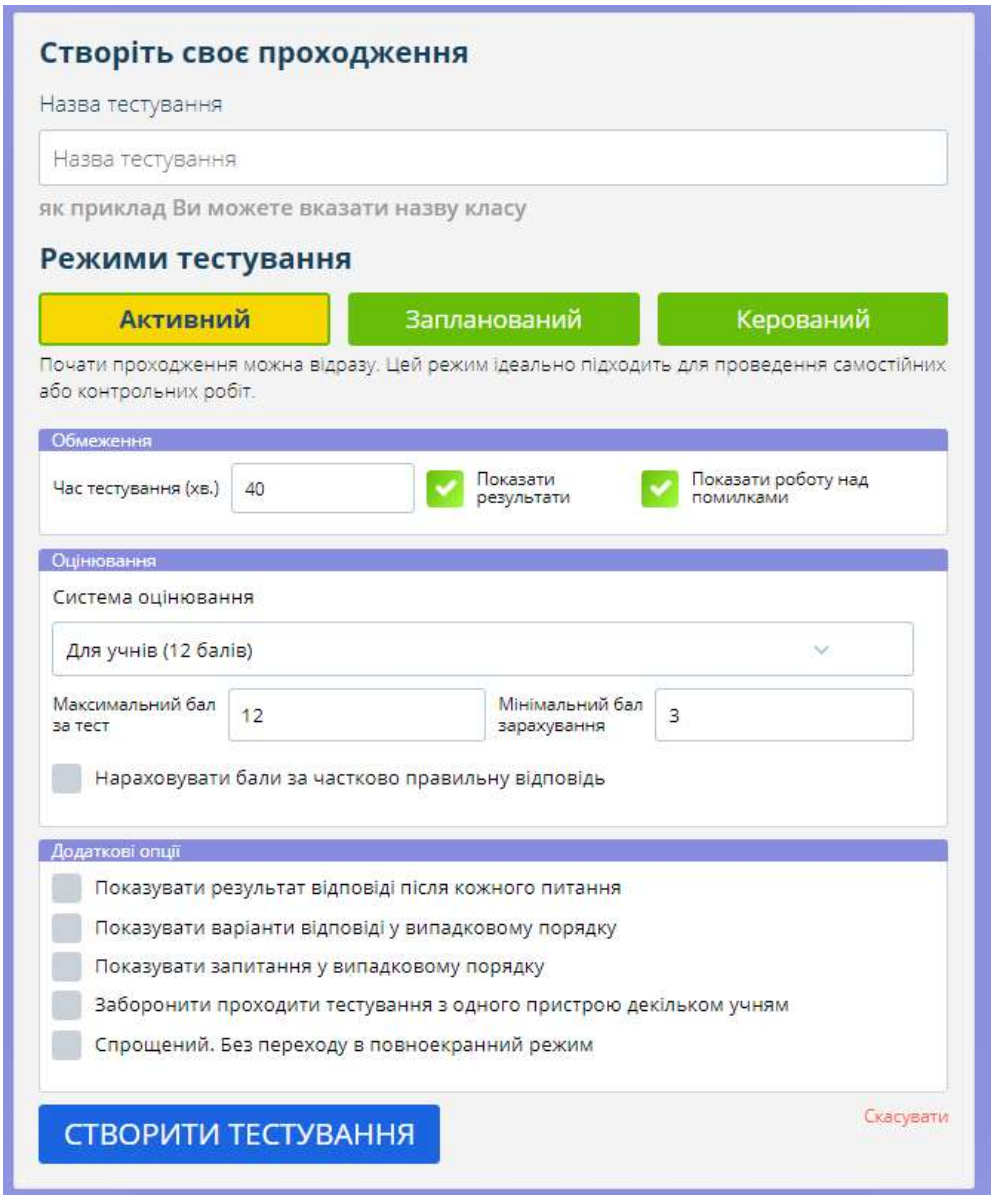


Рис 1.25. Налаштування тесту [9]

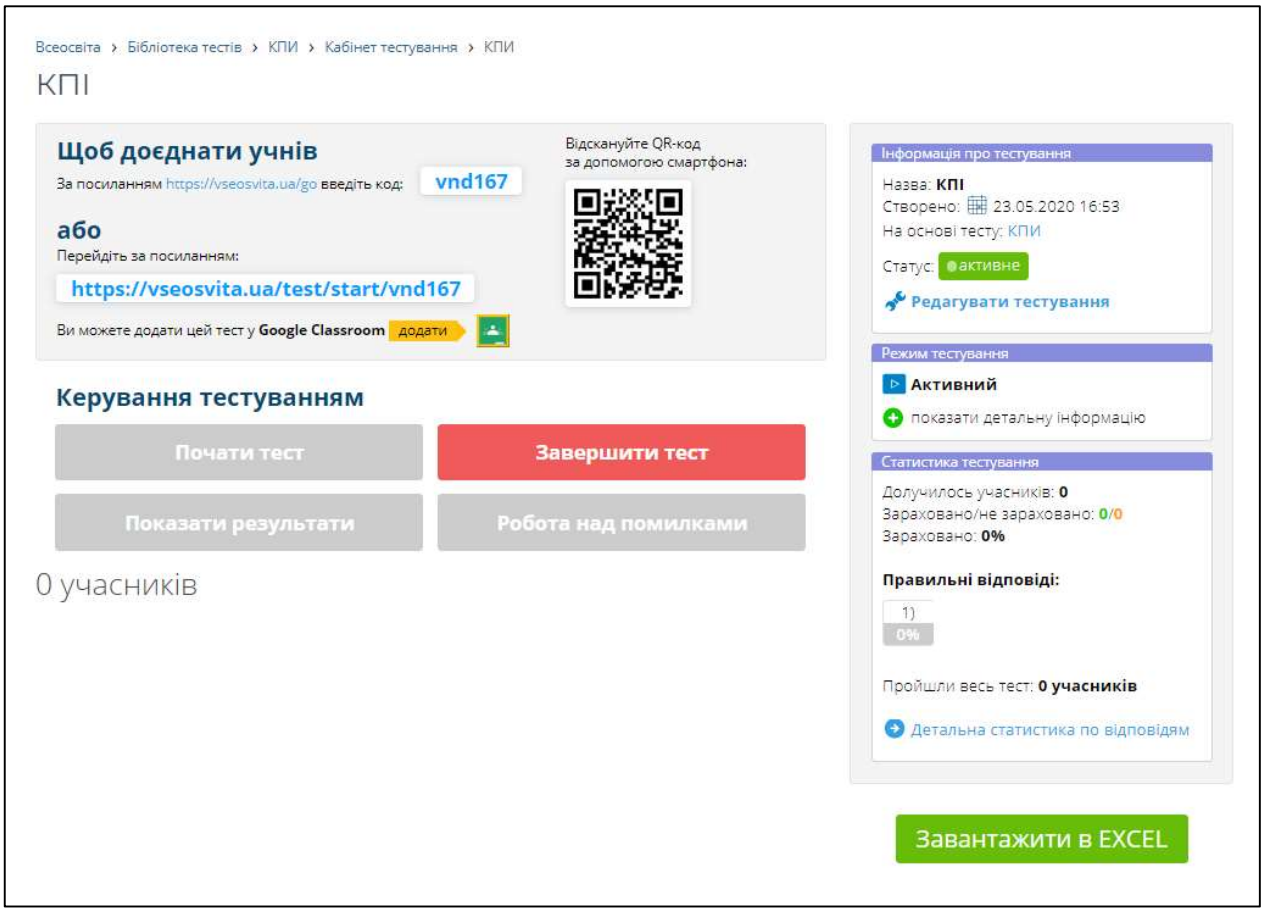


Рис 1.26. Статистика тесту [9]

Тест має дуже гнучкі налаштування(рис 1.25). До налаштуваннь входять: режими тестування (активний, запланований, керований), час тестування, налаштування балу за тест та додаткові опціх такі, як «Заборонити проходити тестування з одного пристрою». Статистика тесту(рис 1.26) відбувається у синхронізованому режимі, також можно побачити кто і скількі разів проходив тест і коли. Тест можливо додати у Google classroom. Для поширювання тесту використовується посилання або QR-код. Статистику тесту можливо завантажити у форматі EXCEL.

Отже, проєкт «Всеосвіта» дуже перспективний, має зручний інтерфейс, багато опцій, багатофункіолнальні тести, багато матеріалів які щодня поповнюються . Я, вважаю, що цей проєкт є найкращим прототипом клієнт-серверної системи підтримки учбового процессу, але має недолік в тому, що не пристосований для використання без інтернету.

ВИСНОВКИ ДО РОЗДІЛУ 1

В даному розділі були наведені системи для підтримки учбового процесу. Ці системи мають свої плюси і мінуси, але більшість з них мають важку регістрацію, і у людини(дитини), не має розуміння як її проходити.

Перераховані недоліки роблять актуальним розробку власної системи, яка буде зручний та простий для навчання та використання, а також зберігати тести для подальшого використання.

РОЗДІЛ 2
ОПИС ПРИНЦИПІВ, ТЕХНОЛОГІЙ ТА МОВ ПРОГРАМУВАННЯ

2.1. MVC та його компоненти

Рішення клієнт серверних систем основане на понятті MVC. MVC (Model – view - controller) (Модель – вигляд - контролер) (Рис.2.1) - це шаблон проєктування, який використовуєься для розробки цих систем.

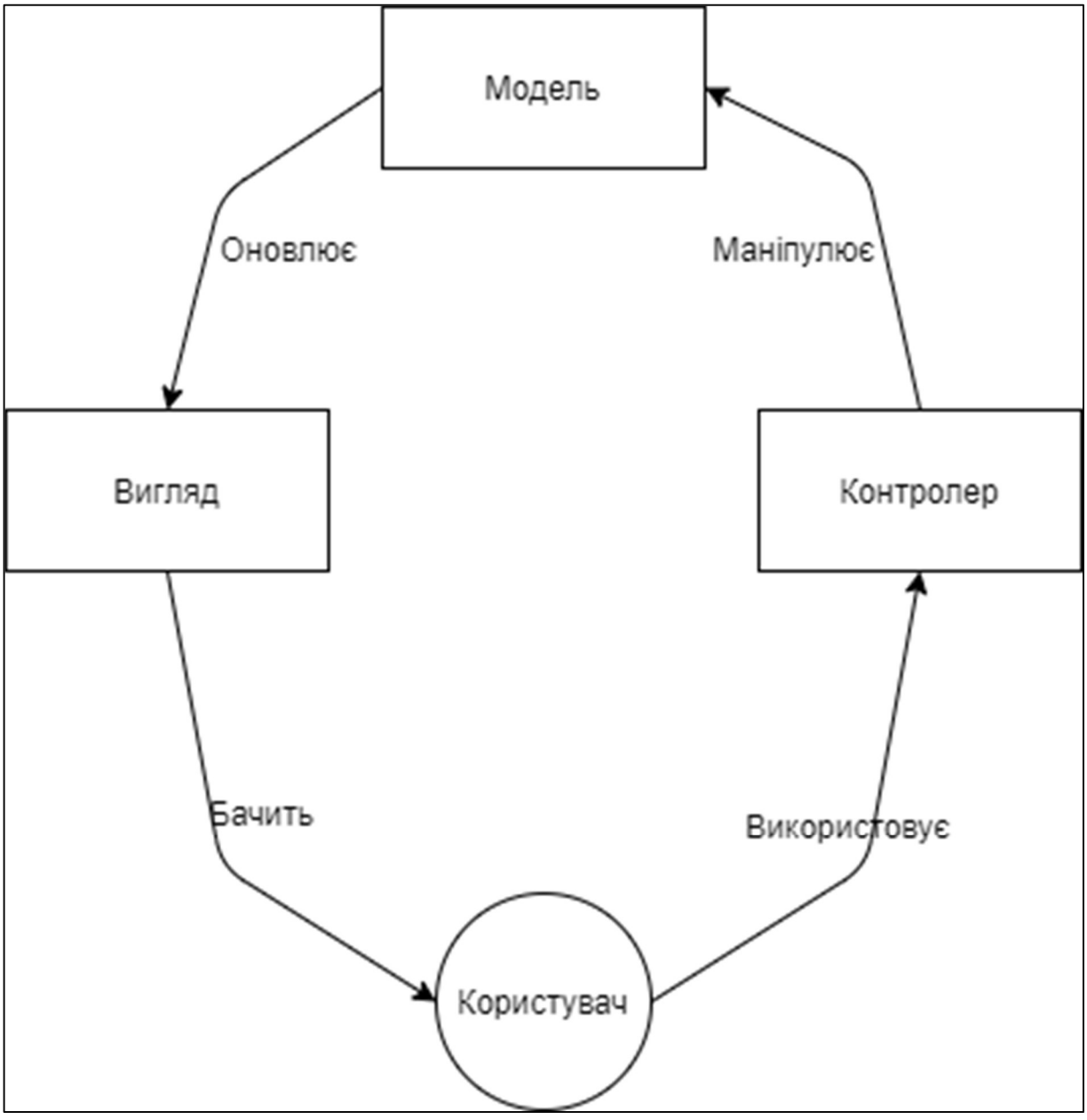


Рис.2.1 MVC

Модель - є центром цього типу проєктування, вона показує поведінку даних, контролює їх логіку та встановлює правила.

Вигляд – це представлення користувачу даними, вже в обробленій формі, за допомогою інтерфейсу, графіків, діаграм.

Контролер – він стоїть між моделлю та виглядом та контролює потоки даних, які входять у застосунок та перетворює на команди.

2.2. Реалізації концепції MVC

Найвідоміші реалізації діляться на такі мови:

- Java
- C#
- Objective-c

Порівняши ці мови, я зупинився на виборі мови Java. Цю концепцію, в цій мові можливо розробити за допомогою Spring Framework.

2.3. Мова програмування Java.

Мови програмування - це штучні мови, створенні для опису команд які передаються ЕОМ. Програмування (кодування) – аналіз алгоритму та транслювання його на мову програмування.

2.3.1. Історія

Історія Java починається з1991 року, коли група інженерів з компанії Sun Microsystems під керівництвом Патріка Нотона (Patrick Naughton) і члена ради директорів (і різнобічного фахівця) Джеймса Гослінга (James Gosling) зайнялася розробкою мови, якою можна було б використовувати для програмування побутових пристроїв, наприклад, контролерів для перемикання каналів кабельного телебачення. Подібні пристрої не володіють великими обчислювальними потужностями і обсягом оперативної пам'яті, і тому новий мову мав бути простим і здатним генерувати дуже компактний код. Крім того, різні виробники можуть вибирати різні процесори для контролерів, тому було дуже важливо не

прив'язуватися до конкретної їх архітектурі. Проєкт створення нової мови отримав кодову назву "Green".

Прагнучи отримати компактний і сумісний із різними платформами код, розробники вирішили створити стерпний мову, здатний генерувати проміжний код для віртуальної машини. Більшість співробітників компанії Sun Microsystems мали досвід роботи з операційною системою Unix, тому в основу розроблюваного ними мови був покладений мову C ++, а не Lisp, Smalltalk або Pascal.

Як сказав Гослінг в своєму інтерв'ю: "Мова - це завжди засіб, а не мета". Спочатку Гослінг вирішив назвати його Oak (Дуб). (Можливо тому, що він любив дивитися на дуб, що ріс прямо під вікнами його робочого кабінету в компанії Sun Microsystems.) Потім співробітники компанії дізналися, що слово "Oak" вже використовується в якості імені раніше створеного мови програмування, і змінили назву на Java. Цей вибір був зроблений з натхнення. [1]

Весь 1993 рік і половину 1994г роки тривали безрезультатні пошуки покупців продукції, розробленої в рамках проєкту "Green", що отримав нову назву - "First Person, Inc.". Патрік Нотон, один із засновників групи, в основному займався маркетингом, налітав в цілому більше 300 тисяч миль, намагаючись продати розроблену технологію. Робота над проєктом "First Person, Inc."була припинена в 1994 р.

Тим часом в рамках Інтернету почала розвиватися система під назвою World Wide Web (Всесвітня павутина). Ключовим елементом цієї системи був браузер, що перетворює гіпертекстові дані в зображення на екрані. У 1994 році більшість користувачів застосовували некомерційний веб-браузер Mosaic, розроблений в суперкомпьютерном центрі університету штату Іллінойс в 1993р.

Сам браузер був розроблений Патріком Нотона і Джонатаном Пейном (Johnatan Payne). Пізніше він був доопрацьований і отримав ім'я HotJava. Щоб продемонструвати всі можливості Java, браузер був написаний на цій

мові. Розробники браузера HotJava наділили свій продукт здатністю виконувати код на небі-сторінках. Програмний продукт, що підтверджував дієвість нової технології, був представлений 23 травня 1995 року в виставці SunWorld '95 і викликав загальний інтерес до Java, що зберігається і донині.

Компанія Sun Microsystems випустила першу версію Java на початку 1996р. Користувачі швидко зрозуміли, що версія Java 1.0 не підходить для розробки серйозних додатків. Звичайно, цю версію можна застосовувати для реалізації візуальних ефектів на веб-сторінках, наприклад, написати аплет, що виводить на сторінку випадково "стрибучий" текст, але версія Java 1.0 була ще сирий. У ній навіть були відсутні кошти виведення на друк. Грубо кажучи, версія Java 1.0 ще не була готова.

У наступній версії, Java 1.1, були усунені найбільш очевидні недоліки, поліпшені засоби рефлексії і реалізована нова модель подій для програмування графічних користувальницьких інтерфейсів (GUI). Але, незважаючи на це, її можливості були все ще обмежені.

Компанія Sun Microsystems випустила першу версію Java на початку 1996 р Користувачі швидко зрозуміли, що версія Java 1.0 не підходить для розробки серйозних програм. Звичайно, цю версію можна застосовувати для реалізації візуальних ефектів на веб-сторінках, наприклад, написати аплет, що виводить на сторінку випадково "стрибучий" текст, але версія Java 1.0 була ще сирий. У ній навіть були відсутні кошти виведення на друк. Грубо кажучи, версія Java 1.0 ще не була готова.

У наступній версії, Java 1.1, були усунені найбільш очевидні недоліки, поліпшені засоби рефлексії і реалізована нова модель подій для програмування графічних користувальницьких інтерфейсів (GUI). Але, незважаючи на це, її можливості були все ще обмежені.

Випуск версії Java 1.2 став основною новиною на конференції JavaOne в 1998 р.

У новій версії слабкі засоби для створення GUI і графічних додатків були замінені потужним інструментарієм. Це був крок вперед, до реалізації гасла

"Write Once, Run Anywhere" ("Одного разу написано -виконуємо всюди"), висунутого при розробці попередніх версій. У грудні 1998 року через три дні після виходу в світ назва нової версії було змінено на громіздке Java 2 Standard Edition Software Development Kit Version 1.2 (Стандартна редакція набору інструментальних засобів для розробки програмного забезпечення на Java 2, версія 1.2).

Крім Standard Edition, були запропоновані ще два варіанти: Micro Edition ("мікроредакція") для портативних пристроїв, таких як мобільних телефонів, і Enterprise Edition (редакція для корпоративних додатків). .

Версії 1.3 і 1.4 редакції Standard Edition стали результатом поетапного удосконалення спочатку випущеної версії Java 2. Вони володіли новими можливостями, підвищеною продуктивністю і, зрозуміло, містили набагато менше помилок. В процесі розвитку Java багато поглядів на аплети і клієнтські програми були переглянуті. Зокрема, виявилося, що на Java зручно розробляти високоякісні серверні додатки.

У версії 5.0 мову Java піддався найбільш істотною модифікації з моменту випуску версії 1.1. (Спочатку версія 5.0 мала номер 1.5, але на конференції

JavaOne в 2004 р була прийнята нова нумерація версій.) Після багаторічних досліджень були додані узагальнені типи (які приблизно відповідають шаблонами C ++), хоча при цьому не були висунуті вимоги модифікації віртуальної машини. Ряд інших мовних елементів, наприклад, цикли в стилі for each, автоупаковка і анотації, були явно "навіяні" мовою C #.

Версія 6 (без суфікса .0) була випущена в кінці 2006 р Знову ж сама мова не зазнав істотних змін, але були внесені удосконалення, пов'язані з продуктивністю, а також проведені розширення бібліотек. У міру того як в

центрах обробки даних все частіше стали застосовуватися апаратні засоби широкого споживання замість спеціалізованих серверів, для компанії Sun Microsystems настали важкі часи, і в кінцевому підсумку в 2009 році вона була придбана компанією Oracle.

Розробка наступних версій Java призупинилася на довгий час. І тільки в 2011 році компанія Oracle випустила нову версію Java 7 з простими вдосконаленнями.

У 2014 році була випущена версія Java 8 з найбільш істотними змінами майже за двадцять років існування цієї мови. У версії Java 8 зроблена спроба впровадити стиль функціонального програмування, щоб спростити вираз обчислень, які можуть виконуватися паралельно. Всі мови програмування повинні розвиватися, щоб залишатися актуальним, і в цьому відношенні мова Java проявив себе з найкращого боку.

Історія головного нововведення у версії Java 9 відноситься ще до 2008 року, коли Марк Рейнгольд (Mark Reinhold), головний розробник платформи Java, зробив зусилля, щоб якимось чином впоратися з громіздкою, монолітною платформою Java. Щоб досягти поставленої мети, довелося впровадити модулі - самостійні блоки коду, що надають конкретні функціональні можливості.

На розробку і реалізацію модульної системи, цілком вписується в платформу Java, пішло одинадцять років, але ще матимуть змогу переконатися, наскільки добре вона підходить для додатків і бібліотек Java. Версія Java 9, випущена в 2017 році, володіє і іншими привабливими мовними засобами, описаними в цьому виданні.

Починаючи з 2018 року, версії Java випускаються через кожні шість місяців, щоб прискорити впровадження нових мовних засобів. Хоча деякі версії (наприклад, Java 11) розраховані на підтримку в довгостроковій перспективі.

2.3.2. Простота

Під Синтаксис Java, по суті, являє собою спрощений варіант синтаксису C ++. У цій мові відсутня потреба в файлах заголовків, арифметиці (і навіть в синтаксисі) покажчиків, структурах, об'єднаннях, перевантаження операцій, віртуальних базових класах і т.п. Але творці Java не прагнули виправити всі недоліки мови C ++. Наприклад, синтаксис оператора switch в Java залишився незмінним. Знаючи C ++, неважко перейти на Java.

Коли був випущений мову Java, C ++ був аж ніяк не найпоширенішою мовою програмування. Багато розробники користувалися мовою Visual Basic і його середовищем програмування шляхом перетягування. Цим розробникам Java здався непростим мовою, тому їм треба було кілька років для оволодіння середовищами розробки на Java. В даний час середовища розробки на Java просунулися далеко вперед в порівнянні з більшістю інших мов програмування.

Інший аспект простоти - стисліcть. Одна з цілей мови Java - забезпечити розробку незалежних програм, здатних виконуватися на машинах з обмеженим обсягом ресурсів. Розмір основного інтерпретатора і засобів підтримки класів становить близько 40 Кбайт; стандартні бібліотеки та засоби підтримки потоків, в тому числі автономне микроядро, займають ще 175 Кбайт.[1]

2.3.3. ООП

Об'єктно-орієнтоване програмування (ООП) в даний час стало домінуючою методикою програмування, витіснивши структурні або процедурні підходи, розроблені в 1970-х роках.

Об'єктно-орієнтована програма складається з об'єктів, кожен з яких володіє певними функціональними можливостями, наданими в розпорядження користувачів, а також прихованої реалізацією. одні об'єкти для своїх програм ви можете взяти в готовому вигляді з бібліотеки, інші вам доведеться спроєктувати самостійно.

Будувати чи свої об'єкти або купувати готові - залежить від вашого бюджету або часу. Але, як правило, до тих пір, поки об'єкти задовольняють вашим вимогам, вам не потрібно особливо турбуватися про те, яким чином реалізовані їх функціональні можливості.

Традиційне структурне програмування полягає в розробці набору процедур (або алгоритмів) для вирішення поставленого завдання. Визначивши ці процедури, програміст повинен знайти підходящий спосіб зберігання даних.

Ось чому творець мови Pascal Ніклаус Вірт (Niklaus Wirth) назвав свою відому книгу з програмування Algorithms + Data Structures = Programs (Алгоритми + Структури даних = Програми). Зверніть увагу на те, що в назві цієї книги алгоритми стоять на першому місці, а структури даних - на другому. Его відображає образ мислення програмістів того часу.

Спочатку вони вирішували, як маніпулювати даними, а потім - яку структуру застосувати для організації цих даних, щоб з ними було легше працювати. Підхід ООП в корені змінив ситуацію, поставивши на перше місце дані і лише на друге - алгоритми, призначені для їх обробки.

Для вирішення невеликих завдань процедурний підхід виявляється цілком придатним. Але об'єкти більш пристосовані для вирішення більш великих завдань. Реалізація веб-браузер може зажадати 2000 процедур, кожна з яких маніпулює набором глобальних даних.

У стилі ООП та ж сама програма може бути складена за все з 100 класів, в кожному з яких в середньому визначено по 20 методів. Така структура програми набагато зручніше для програмування, і в ній легше знаходити помилки. Припустимо, дані деякого об'єкту знаходяться в неправильному стані. Очевидно, що набагато легше знайти причину неполадок серед 20 методів, що мають доступ до даних, чим серед 2000 процедур.

2.3.4. Об’єктно-орієнтована спрямованість

Об'єктно-орієнтований підхід до програмування цілком усталився на момент появи мови Java. Дійсно, особливості Java, пов'язані з об'єктами,

можна порівняти з мовою C ++. Основна відмінність між ними полягає в механізмі множинного спадкоємства, який в Java замінений більш простим поняттям інтерфейсів. Мова Java володіє великими можливостями для самоаналізу при виконанні, ніж C ++.

2.3.5. Автоматичне керування пам’яттю

Хороші мови - не рідкість, а поява деяких з них викликало свого часу справжню сенсацію в обласгі обчислювальної техніки. На відміну від них, Java - це програмна платформа, що включає в себе потужну бібліотеку, великий обсяг коду, придатного для повторного використання, а також середовище для виконанняпрограм, яка забезпечує безопасносгь, незалежність від операційної сісгеми і автоматичну збірку "сміття".

Программісгам потрібні мови з чіткими синтаксичними правилами і зрозумілою семантикою (тобто безумовно не C ++). Такому вимогу, крім Java, відповідають десятки мов. Деякі з них навіть забезпечують переносімосгь і збірку "сміття", але їх бібліотеки осгавляют бажати багато кращого.

В результаті программісги змушені самосгоятельно реалізовувати графічні операції, доступ до мережі і базі даних і інші часго зустрічаються процедури. Java об'єднує в собі прекрасну мову, висококачесгвенную середу виконання програм і велику бібліотеку. В результаті багато программісги осгановілі свій вибір саме на Java.

2.3.6. Динамічні, переносимість

На відміну від C і C ++, жоден з аспектів специфікації Java не залежить від реалізації. Розрядність примітивних типів даних і арифметичні операції над ними строго визначені.

Наприклад, тип int в Java завжди означає 32-розрядний ціле число, а в C і C ++ тип int може означати як 16-, так і 32-розрядний ціле число. Єдине обмеження полягає в тому, що розрядність типу int не може бути менше розрядності типу short int і більше розрядності типу long int. Фіксована

розрядність числових типів даних дозволяє уникнути багатьох неприємностей, пов'язаних з виконанням програм на різних комп'ютерах.

Двійкові дані зберігаються і передаються в незмінному форматі, що також дозволяє уникнути непорозумінь, пов'язаних з різним порядком проходження байтів на різних платформах. Символьні рядки зберігаються в стандартному форматі Юнікод.

Бібліотеки, які є частиною системи, надають переносяться інтерфейси. Наприклад, в Java передбачений абстрактний клас Window і його реалізації для операційних систем Unix, Windows і Macintosh ".

Приклад класу Window, мабуть, був обраний не зовсім вдало. Всякий, коли-небудь намагався написати програму, яка однаково добре працювала б під управлінням операційних систем Windows, Травні OS і десятка різновидів ОС Unix, знає, що це дуже важке завдання. Розробники версії Java 1.0 зробили героїчну спробу вирішити цю задачу, надавши простий набір інструментальних засобів, пристосовуватися звичайні елементи призначеного для користувача інтерфейсу до різних платформ.

На жаль, в кінцевому підсумку вийшла бібліотека, працювати з якою було непросто, а результати виявлялися чи прийнятними в різних системах. Цей початковий набір інструментів для побудови призначеного для користувача інтерфейсу був у подальшому не раз змінений, хоча переносимість програм на різні платформи, як і раніше залишається проблемою.

Проте з усіма завданнями, які не мають відношення до призначених для користувача інтерфейсів, бібліотеки Java відмінно справляються, дозволяючи розробникам працювати, не прив'язуючись до конкретної платформі. Зокрема, вони можуть користуватися файлами, регулярними виразами, XML-розміткою, датами і часом, базами даних, мережевими з'єднаннями, потоками виконання та іншими засобами, що не спираючись на базову операційну систему.

Програми на Java не тільки стають переносяться, але і прикладні програмні інтерфейси Java API нерідко виявляються більш високої якості, ніж їх переносних орієнтовані аналоги. Зазвичай інтерпретується байт-код має достатню продуктивність, але бувають ситуації, коли потрібно ще більш висока продуктивність.

Байт-код можна транслювати під час виконання програми в машинний код для того процесора, на якому вьтолняется цю програму.

На ранній стаді розвитку Java багато користувачів були не згодні з твердженням, що продуктивності "більш ніж достатньо". Але тепер динамічні компілятори (звані інакше JIT-компіляторами) настільки вдосконалені, що можуть конкурувати з традиційними компиляторами, а в деяких випадках вони навіть дають виграш в продуктивності, оскільки мають більше доступною інформації.

Так, наприклад, динамічний компілятор може відстежувати код, який виконується частіше, і оптимізувати по швидкодії тільки цю частину коду. Динамічному компілятору відомо, які саме класи були завантажені. Він може спочатку застосувати вбудовування, коли деяка функція взагалі не переопределяется на підставі завантаженої колекції класів, а потім скасувати, якщо буде потрібно, таку оптимізацію.

2.3.7. Інтерпретованість

Інтерпретатор Java може виконувати байт-код безпосередньо на будь-який мащін, на яку перенесено інтерпретатор. А оскільки процес компонування носить в большеей ступеня покроковий і відносно простий характер, процес розробки програм може бути помітно прискорений, ставши більш творчим.

З цим можна погодитися, але з великою натяжкою. Всім, хто має досвід програмування на Lisp, Smalltalk, Visual Basic, Python, R або Scala, добре відомо, що насправді означає "прискорений і більш творчий" процес розробки.

Випробовуючи що-небудь, ви відразу ж бачите результат. Але такий досвід просто відсутня в роботі із середовищами розробки на Java. Але так було до версії Java 9, коли з'явилося інструментальне засіб jshell, що підтримує швидке експериментальне програмування в діалоговому режимі.

2.3.8. Продуктивність

Зазвичай інтерпретується байт-код має достатню продуктивність, але бувають ситуації, коли потрібно ще більш висока продуктивність.

Байт-код можна транслювати під час виконання програми в машинний код для того процесора, на якому вьтолняется цю програму.

На ранній стаді розвитку Java багато користувачів були не згодні з твердженням, що продуктивності "більш ніж достатньо". Але тепер динамічні компілятори (звані інакше JIT-компіляторами) настільки вдосконалені, що можуть конкурувати з традиційними компиляторами, а в деяких випадках вони навіть дають виграш в продуктивності, оскільки мають більше доступної інформації.

Так, наприклад, динамічний компілятор може відстежувати код, який виконується частіше, і оптимізувати по швидкодії тільки цю частину коду. Динамічному компілятору відомо, які саме класи були завантажені. Він може спочатку застосувати вбудовування, коли деяка функція взагалі не переопределяется на підставі завантаженої колекції класів, а потім скасувати, якщо буде потрібно, таку оптимізацію.

2.3.9. Безпека

Нижче перераховані деякі види порушення захисту, які з самого початку забезпечує захис Java.

- Навмисне переповнення стека виконуваної програми - один з розповсюджених способів порушення захисту, використовуваних вірусами і "хробаками".
- Пошкодження даних на ділянках пам'яті, що знаходяться за межами простору, виділеного процесу.

- Несанкціоноване читання файлів і їх модифікація.

Спочатку в Java було прийнято досить відповідальне ставлення до завантажуваного коду. Ненадійний і безпечний код виконувався в середовищі "пісочниці", де він не міг чинити негативного впливу на головну систему. Користувачі могли бути впевнені, що в їх системі не станеться нічого поганого через виконання коду Java незалежно від його походження, оскільки він просто не міг проникнути з "Пісочниці" назовні.

Але модель безпеки в Java складна. Незабаром після випуску першої версії Java Development Kit група фахівців з безпеки з Прінстонського університету виявила в системі безпеки ледь помітні програмні помилки, які дозволяли здійснювати атаки на головну систему з ненадійного коду.

Ці помилки були швидко виправлені. Але, на жаль, зловмисники примудрилися знайти незначні прогалини в реалізації архітектури безопасносгі. І компанії Sun Microsystems, а потім і компанії Oracle довелося витратити чимало часу на усунення подібних дір.

Після цілого ряду великомасштабних атак виробники браузерів і фахівці з компанії Oracle стали обачнішими. Тепер модулі Java, що підключаються до браузерів, більше не довіряють віддаленого коду, якщо відсутній цифровий підпис цього коду і згоду користувачів на його виконання.

2.3.10. Надійність

Компілятор Java виявляє такі помилки, які в інших мовах виявляються тільки на етапі виконання програми. Крім того, програмісти, які витратили багато годин на пошуки помилки, що викликала порушення даних в пам'яті через невірного покажчика, будуть раді тому, що в роботі з Java подібні ускладнення не можуть навіть в принципі виникнути.

2.3.11. Динамічність

Мова Java призначена для створення програм, які працюють в розподіленому середовищі Internet на базі протоколів TCP/IP. Це дуже важливо в тих випадках, коли потрібно додати код в уже поточну програму.

Яскравим тому прикладом служить код, що завантажується з Інтернету для виконання браузером. Зробити це на C або C ++ не так-то просто, але розробники Java були добре обізнані про динамічні мовах програмування, які дозволяли без особливих зусиль розвивати поточну програму. Їх досягнення полягало в тому, що вони впровадили таку можливість в основну мову програмування.

2.3.12. Багатопоточність

В даний час все більшого значення набуває розпаралелювання виконуваних завдань, оскільки дія закону Мура добігає кінця. У нашому розпорядженні тепер є не більше швидкодіючі процесори, а більше їх кількість, і тому ми повинні завантажити їх корисною роботою, щоб вони не простоювали. Але, на жаль, в більшості мов програмування проявляється вражаюче нехтування цією проблемою.

І в цьому відношенні Java випередив свій час. Він став першим з основних мов програмування, де помержівалось паралельне програмування. Як випливає з утюмянутого вище офіційного опису Java, спонукальна причина до такої підтримки була дещо іншою. У той час багатоядерні процесори були рідкістю, а хмарне-програмування тільки починало розвиватися, і тому процесорам доводилося довго простоювати в очікуванні відповіді від сервера.

Паралельне програмування потрібно для того, щоб інтерфейс користувача не застигав в очікуванні подібніх відповідей. І хоча паралельне програмування ніколи не було простим заняття, в Java зроблено чи мало, щоб цей процес став більш керованого.

2.3.13. Колекції

У початковій версії Java пропонувався лише невеликий набір класів для найбільш уживаних структур даних: Vector, Stack, Hashset, BitSet, а також інтерфейс Enumeration, предосивлявшій абстрактний механізм для звернення до елементів, що знаходяться в довільному контейнері. Безумовно, це було

мудре рішення, адже для реалізації всеосяжної бібліотеки класів колекцій потрібен час і досвід.

Після випуску версії Java 1.2 розробники усвідомили, що настав час створити повноцінний набір структур даних. При цьому вони зіткнулися з багатьма суперечливими вимогами. Вони прагнули до того, щоб бібліотека була компактною і простий в освоєнні.

Для цього потрібно було уникнути складності стандартної бібліотеки шаблонів (STL) в C ++, але в той же час запозичити узагальнені алгоритми, вперше з'явилися в бібліотеці STL. Крім тою, потрібно було забезпечити сумісність успадкованих класів колекцій з новою архітектурою. Як це трапляється з усіма розробниками бібліотек колекцій, їм доводилося не раз робити нелегкій вибір, знаходячи попутно чимало оригінальних рішень.

2.4. Spring Framework – як інструмент для мови Java

2.4.1. Історія

За Spring Framework стоїть цікава історія. До Spring Framework програми були розроблені за допомогою традиційних стандартів JEE. Ці стандарти надавали деякі дуже чудові функції, такі як управління транзакціями, обмін повідомленнями, розсилка, інтерфейс каталогів тощо, але оскільки нічого не відбувається без оплати витрат, у нього також були деякі недоліки, такі як:

1. Написання коду було дуже складним, оскільки під час написання компонента потрібно писати набір XML-файлів, домашніх інтерфейсів, віддалених / локальних інтерфейсів тощо.

2. Кожен раз, коли компонент залежав від іншого компонента, він повинен шукати компоненти, від яких він залежав сам. Цей компонент "перегляд" відбувається лише за назвою, тому ім'я залежності було жорстко закодовано в компоненті.

3. Оскільки всі функції, такі як кластеризація, видалення тощо, підтримувались, ви повинні їх налаштувати, незалежно від того, потрібні вони вам чи ні. Це зробить ваші програми роздутими.

Усі ці проблеми були вирішені впровадженням Spring Framework. У жовтні 2002 року Род Джонсон, австралійський спеціаліст з комп'ютерів, написав книгу під назвою "Експерт" Один на один "Проєктування та розробка J2EE".

У цій книзі він запропонував більш просте рішення, засноване на звичайних Java-класах (POJO) та введенні залежності. Він написав понад 30 000 рядків інфраструктурного коду, який включав ряд java інтерфейсів для багаторазового використання та класи для розробки програми. Приблизно в лютому 2003 року Род, Юрген та Ян почали співпрацювати над проєктом Spring. Назва " Spring "(Весна) була дана, оскільки означала новий початок після "Winter"(зими) традиційного J2EE.

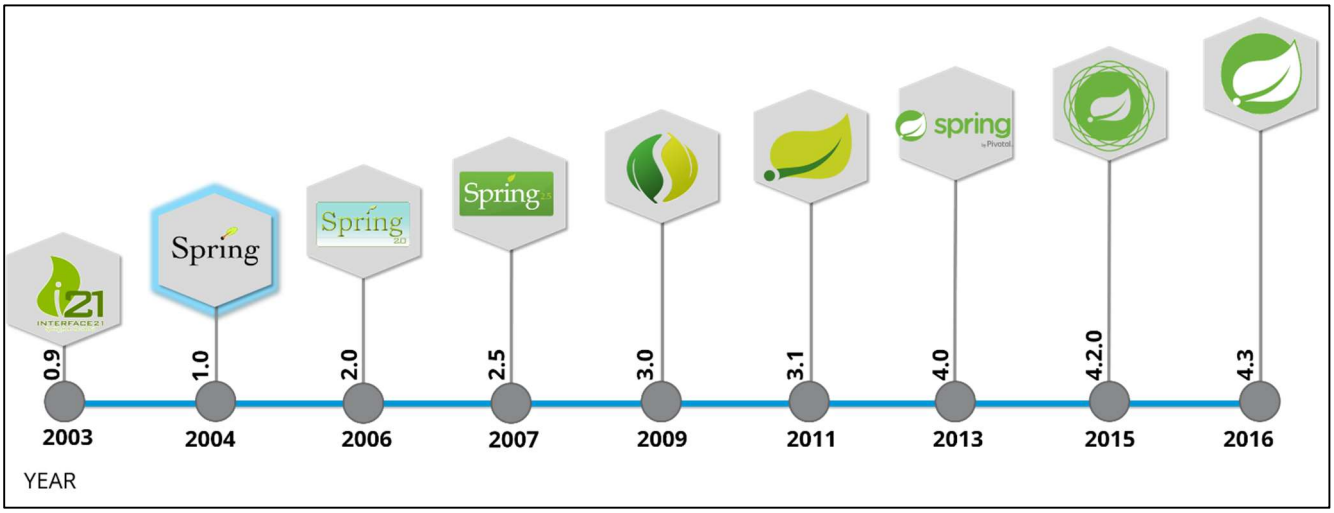


Рис.2.2 Історія Spring Framework[4]

2.4.2. Принципи: Dependency Injection (введення залежності) та Inversion of control (інверсії контролю).

Інверсія управління - це принцип в інженерії програмного забезпечення, за допомогою якого управління об'єктами або частинами програми передається контейнеру або рамці. Найчастіше використовується в контексті об'єктно-орієнтованого програмування.

На відміну від традиційного програмування, в якому наш користувацький код здійснює дзвінки до бібліотеки, IoC дозволяє рамці взяти під контроль потік програми та здійснити дзвінки до нашого

користувацького коду. Щоб увімкнути це, фреймворки використовують абстракції з додатковою поведінкою, вбудованою. Якщо ми хочемо додати власну поведінку, нам потрібно розширити класи фреймворку або підключити власні класи.

Перевагами цієї архітектури є:

- відокремлення виконання завдання від його виконання
- полегшує перемикання між різними реалізаціями
- більша модульність програми
- більша легкість у тестуванні програми, виділяючи компонент або висміюючи його залежності та дозволяючи компонентам спілкуватися через контракти

Інверсія управління може бути досягнута за допомогою різних механізмів, таких як: схема дизайну стратегії, модель локатора обслуговування, заводський зразок та введення залежності (DI).

Введення залежності - це модель, за допомогою якої можна реалізувати IoC, де керований елемент керування - це встановлення залежності об'єкта.

Акт з'єднання об'єктів з іншими об'єктами або "впорскування" об'єктів в інші об'єкти здійснюється асемблером, а не самими об'єктами.

І IoC, і DI - це прості поняття, але вони мають глибокий вплив на те, як ми структуруємо наші системи, тому вони цілком варто добре розуміти.

2.4.3. Spring IoC Container

Контейнер IoC є загальною характеристикою рамок, які реалізують IoC. У рамках Spring контейнер IoC представлений інтерфейсом ApplicationContext. Контейнер Spring відповідає за інстанціювання, налаштування та збирання об'єктів, відомих як квасоля, а також за керування їх життєвим циклом.

Spring framework забезпечує кілька реалізацій інтерфейсу ApplicationContext - ClassPathXmlApplicationContext і

FileSystemXmlApplicationContext для автономних програм та WebApplicationContext для веб-додатків.

Для збирання бобів контейнер використовує метадані конфігурації, які можуть мати форму конфігурації XML або анотації.

2.5. База даних

База даних – це дані які зібрадні в одному місті, та характеризуються концепцією, яка описує ці дані і звязки між ними. У базах дані представляються у вигляді таблиць, також вони містять схеми, подання, процедури та інші об’єкти.

Система керування базамми даних(СКБД)(рис. 2.3) – це система, яка забезпечує створення, маніпулювання, контроль, керування та використання баз данних.

Рис.2.3 Найвідоміші СКБД

Архітектура СКБД:

- Архітектура «файл-сервер» - ця архітектура базується на тому, що одна з машин мережі, буде використовуватися як сервер.
- Архітетура «клієнт-сервер» - ця архітектура коли в мережі існує сервер баз даних, ним виступає комп’ютер, в якому встановленні ці бази даних та програмне забезпечення для цих баз. Клієнти, для

використання ції баз, виконують запити на сервер. Сервер опрацьовує запити та надсилає відповіді.

Види баз даних:

- Ієрархічна – дані в цій базі представляються як дерево, яке складається з об’єктів та звязків між ними. У об’єкта-child лише один parent.
- Мережева – дані в цій базі структоровані як в ієрархічної, але у об’єкта-child може буде багато parent.
- Реляційна – дані в цій базі зберігаються у виді таблиць.
- Об’єкто-орієнтована – дані в цій базі зберігаються у вигляді моделей об’єктів

Бази даних також поділяються на SQLта NoSQL.

SQL (*Structured query language* — мова структурованих запитів) – мова для реаліаційних баз даних.

Різниця SQLта NoSQL:

- Структура і тип даних – в реаліаційних базах, дані повинні бути однозначно певної структури, а в NoSQL ціх обмежень немає
- Запити – реаліаційні використовують SQL-стандарти, тому для роботи з ними використовують мову SQL. Так як NoSQL у свою чергу для кожної своєї бази може використовувати свій спосіб
- Маштабування – обидва типи баз даних добре маштабуються вертикально, але горизонтальне краще у NoSQL.
- Надійнійність – краща надійність у реаляційних баз.
- Підтримка – SQL бази даних мають довгу історію та вони дуже популярні, тому знайти вирішення проблеми завжи легко, що не можна сказати про нові бази як наприклад NoSQL база MongoDB.

2.6. Системи автоматизації побудови проєкту

Автоматична побудова – це етап проєктування, коли відбувається написання скрипту або спектр завдань, які розробники виконують в повсякденій діяльності.

Дії для автоматизації побудови:

- компіляція коду в бінарний,
- збір бінарного коду,
- запуск та проходженя усіх тестів,
- розгортання проєкту на платформі,
- документація.

Найвідоміші рішення для цих систем(рис 2.4):

- Apache Ant
- Apache Maven
- Gradle

Рис.2.4 Найвідоміші системи побудови[3]

2.6.1. Apache Ant

Apache Ant (ant — мураха і водночас акронім — «Another Neat Tool») – java-утиліта для автоматичного збирання проєкту. Платформонезалежний, потребує мову Java, тому краще пристосований для проєктів на мові Java.

Ant використовує XML для описування як проєкт збирати та які залежності він використовує.

Неповний перелік завдань:

- Javac
- Copy
- Delete
- Move
- JUnit
- Zip

2.6.2. Apache Maven

Apache Maven – фреймворк для автоматизованого складання проєкту (рис. 2.5). Створений в 2002 році і на цей час дуже популярний. Кардинально відрізняється від Apache Ant, але також має XML формат для налаштування з більш зручним та простішим виглядом. Maven використовує для опису залежностей компонентів та порядок побудови конструцію Project Object Model.

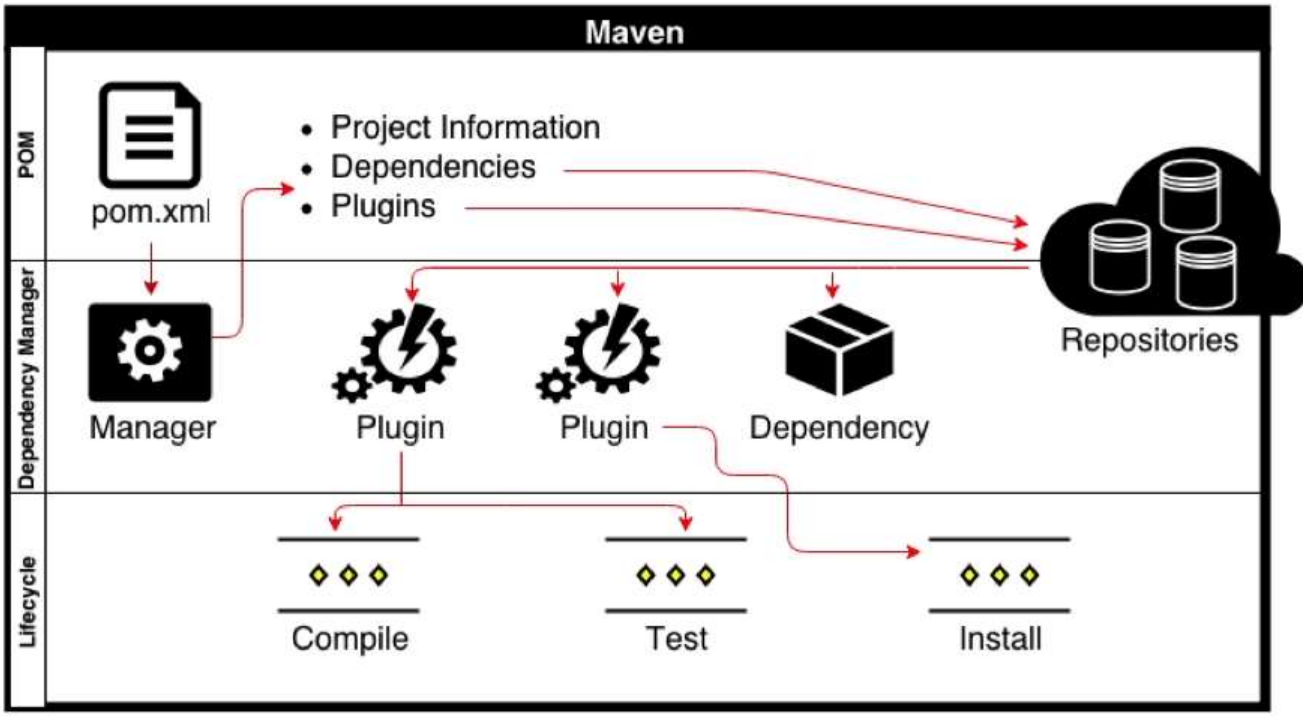


Рис.2.6 Структура Apache Maven[2]

Житевий цикл:

- clean
- default
    - validate
    - test
    - package
    - install
    - deploy
    - compile
    - verify
- site

2.6.3. Gradle

Gradle – система автоматизації збирання проєкту, яка взяла принципи з Apache Maven, Apache Ant та замінивши традиційну XML форму на мову Groovy. Ця система використовує спрямований ациклічний граф для упорядкування завдання, а не життєвий цикл як у попередників. Gradle був розроблений для розробки мультипроєктів, які можуть збільшуватись і підримувати інкрементне збирання проєктів.

Система емітує роботу життєвого циклу Maven, за допомогою графа залежностей для завдань та їх входів і виходів.

Наприклад завдання build використовує:

- compileJava
- jar
- classes
- assemble
- test
- build
- check
- compileTestJava

2.7. Система керування версіями файлів (Git)

Git – широковідома система для керування файлами та їх версіями. Підходить як для маленьких так і для багатомаштабних проєктів. Спочатку проєкт був створений для управління розробкою ядра Linux, але переріс у дещо маштабне.

Нище наведені ринцип роботи системи (Git):

- Збереження файлів – система не зберігає інформацію про файли як список змін, а зберігає у вигляді зліпків. Коли проходить фіксація версія проєкту, Git робить зліпок файлів, якщо файл не змінився, то система перенаправляє на файл, який був раніше збережений.
- Локальні операції – Git можливо використовувати без інтернету, тобто локально. Все зберігається в історію і при потребі можливе вивантаження в репозиторій, який віддалений.
- Цілісність данних – файли в системі всі хешуються за допомогою функції SHA-1 і цей хеш стає індексом цього файлу. Тому Git може з легкістю слідкувати за змінами в файлах.
- Галуження(гілки) – це віділення від основної гілки розробки. У системі можливе створення декількох гілок та вмикати потрібну. За допомогою галуження, можливе багатозадачну розробку проєкту не чіпаючи основну лінію розрубки. Основна гілка за замоучуванням має ім’я master. Гілка є простим файлом.
- Зливання та перебазовування данних – для того щоб об’єднати всі напрацювання та злити їх в одне ціле система використовує: merge (зливання) та rebase (перебазування). Різниця між ними в тому, що зливання просто об’єднує дві гілки, а перебазування перемотує гілку .

ВИСНОВКИ ДО РОЗДІЛУ 2

Виходячи з цього розділу, можна розробка клієнт серверних ситем з використанням принципу MVC було, є і буде актуальним ще довго. Також ми побачили, що Spring Framework є вдалим рішеням для побудови цієї системи.

Розробка цієї системи є актуальною та потрібною для тих людей, які навчаються у школі та не мають достатніх знань володіння комп'ютером та Інтерентом.

Згідно статистиці найпопулярніші мови, мова програмування Java зараз у топі популярних. Тому що вона відносно проста та легка, в ній добре реалізований об'єктно-орієнтований принцип, вона має достатній рівень безпеки навіть для комерційних проєктів. Java використовує автоматичне керування пам'ятю, що слугує великим плюсом. В таких мовах, як C та C++ автоматичне керування пам'ятю відсутнє, і в великих застосунках можливий "витік" пам'яті, який дуже складно локалізувати та виправити. Отже, для побудови системи була обрана Java.

Також я зупинив свій вибір на дуже відомову Spring Framework. За допомогою своїх інструментів, може бути побудовано веб-застосунок або клієнт-серверну систему.

РОЗДІЛ 3

ІНСТРУКЦІЯ КОРИСТУВАЧА

Щоб почати роботу з системою, потрібно налаштавутати базу даних MySql з відповідними скриптами які забражені на рисунках 3.1,3.2

```
CREATE TABLE EXAM (
  ID          INT          NOT NULL AUTO_INCREMENT PRIMARY KEY,
  NAME        VARCHAR(50)  NOT NULL,
  DESCRIPTION VARCHAR(250) NOT NULL
);

CREATE TABLE QUESTION (
  ID           INT          NOT NULL AUTO_INCREMENT PRIMARY KEY,
  EXAM_ID      INT          NOT NULL,
  NAME         VARCHAR(250) NOT NULL,
  MULTI_ANSWER BIT          NOT NULL DEFAULT 0
);

ALTER TABLE QUESTION ADD FOREIGN KEY (EXAM_ID) REFERENCES EXAM (ID);

CREATE TABLE ANSWER (
  ID          INT          NOT NULL AUTO_INCREMENT PRIMARY KEY,
  QUESTION_ID INT          NOT NULL,
  NAME        VARCHAR(250) NOT NULL,
  IS_CORRECT  BIT          NOT NULL DEFAULT 0
);
ALTER TABLE ANSWER ADD FOREIGN KEY (QUESTION_ID) REFERENCES QUESTION (ID);

CREATE TABLE USERS (
  ID       INT         NOT NULL AUTO_INCREMENT PRIMARY KEY,
  USERNAME VARCHAR(50) NOT NULL,
  PASSWORD VARCHAR(50) NOT NULL,
  ENABLED  BIT         NOT NULL DEFAULT 1
);

CREATE TABLE AUTHORITIES (
  ID        INT         NOT NULL AUTO_INCREMENT PRIMARY KEY,
  USERNAME  VARCHAR(50) NOT NULL,
  AUTHORITY VARCHAR(50) NOT NULL
);
ALTER TABLE AUTHORITIES ADD FOREIGN KEY (USERNAME) REFERENCES USERS (USERNAME);

CREATE TABLE PROTOCOL (
  ID              INT      NOT NULL AUTO_INCREMENT PRIMARY KEY,
  EXAM_ID         INT      NOT NULL,
  USER_ID         INT      NOT NULL,
  START           DATETIME NOT NULL,
  FINISH          DATETIME,
  QUESTION_COUNT  INT,
  CORRECT_ANSWERS INT,
  GRADE           INT
);
ALTER TABLE PROTOCOL ADD FOREIGN KEY (EXAM_ID) REFERENCES EXAM (ID);
ALTER TABLE PROTOCOL ADD FOREIGN KEY (USER_ID) REFERENCES USERS (ID);

```

Рис. 3.1 Скрип для створення таблиць

```sql
-- Exams
INSERT INTO EXAM (name, description) VALUES ('Java Exam', 'Show your Java skills!');


-- Java Questions with Answers
INSERT INTO QUESTION (exam_id, name, multi_answer) VALUES (1, 'Which collection cannot contain duplicate elements', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (1, 'Set', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (1, 'Map', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (1, 'List', 0);

INSERT INTO QUESTION (exam_id, name, multi_answer) VALUES (1, 'What are the basic interfaces of Java Collections Framework ?', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (2, 'Set', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (2, 'Random', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (2, 'List', 1);

INSERT INTO QUESTION (exam_id, name, multi_answer) VALUES (1, 'What's a deadlock ?', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (3, 'A condition that occurs when two processes are waiting for each other to complete', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (3, 'Endless loop', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (3, 'Broken lock', 0);


INSERT INTO QUESTION (exam_id, name, multi_answer) VALUES (1, 'How to write a for loop?', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (4, 'for (i <- 0 until 10) { ... }', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (4, 'for (int i=0; i < 10; i++) { ... }', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (4, 'Java have no for loops', 0);

INSERT INTO QUESTION (exam_id, name, multi_answer) VALUES (1, 'What is Java?', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (5, 'Programming language', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (5, 'Teatro La Scala, Milano, Italy', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (5, 'The Rock in Russia', 1);

INSERT INTO QUESTION (exam_id, name, multi_answer) VALUES (1, 'Can I use existing C# code from Java?', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (6, 'Yes', 1);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (6, 'No', 0);
INSERT INTO ANSWER (question_id, name, is_correct) VALUES (6, 'Only starting from Java 8', 0);

```

Рис 3.2. Скрипт для заповнювання таблиць

На рисунку 3.2 система заповнюється питаннями та відповідями на них. Після створення таблиць , ми можемо запускати нашу систему. Потрібно у проєкті системи зібрати .jar файл та запустити.

Після цих маніпуляцій ми переходимо у браузер та бачимо привітальну сторінку(рис. 3.3), у яку потрібно ввести логін та пароль які уже відомі та були видані.

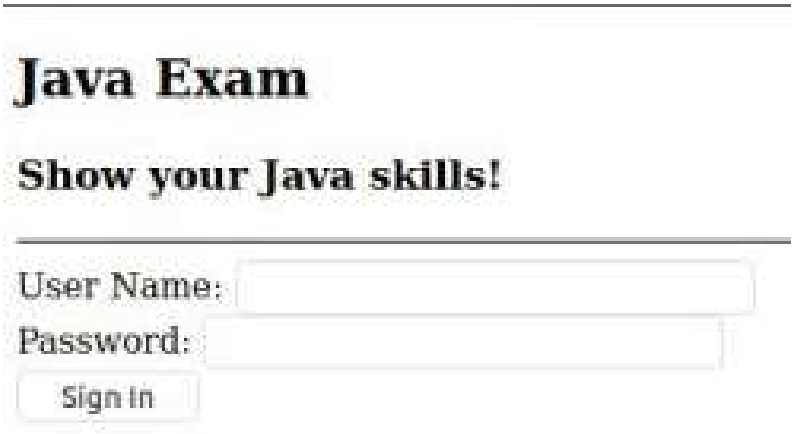


Рис 3.3 Привітальна сторінка

Після успішної авторизації розпочинається тест. З обмеженням по часу. Тест має декілька відповідей(рис. 3.5) або одну(рис.3.4). Також можливо

перейти на інше питання(рис 3.6). Після закінчення часу, або користувач відповів на всі питання , користувачу демонструється його результат (рис. 3.7).

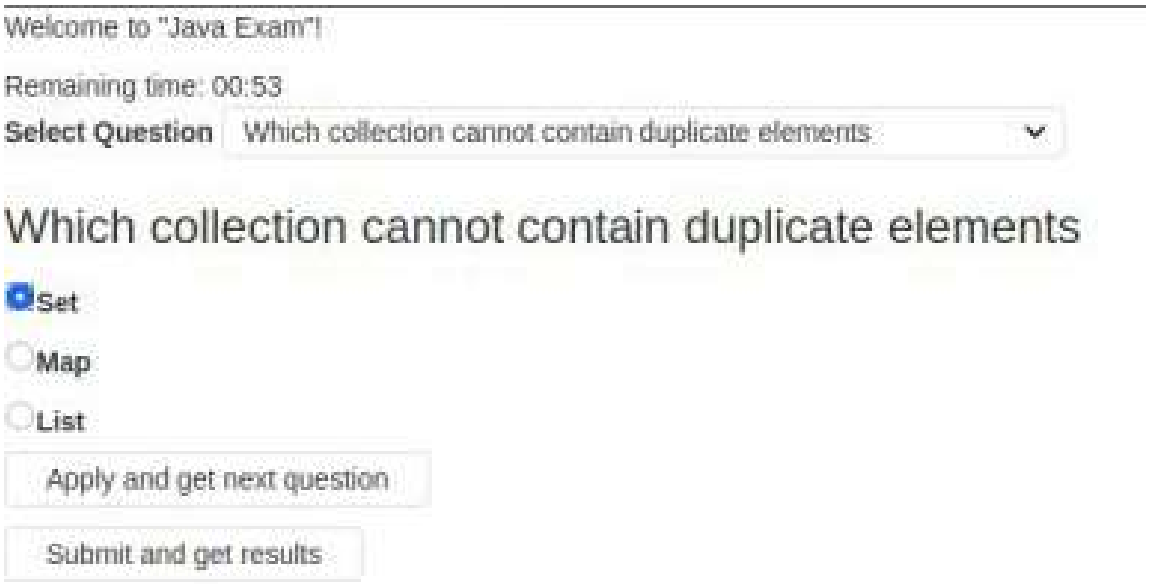


Рис 3.4 Питання з одною правильною відповідю

Рис 3.5 Питання з декількома правильними відповідями

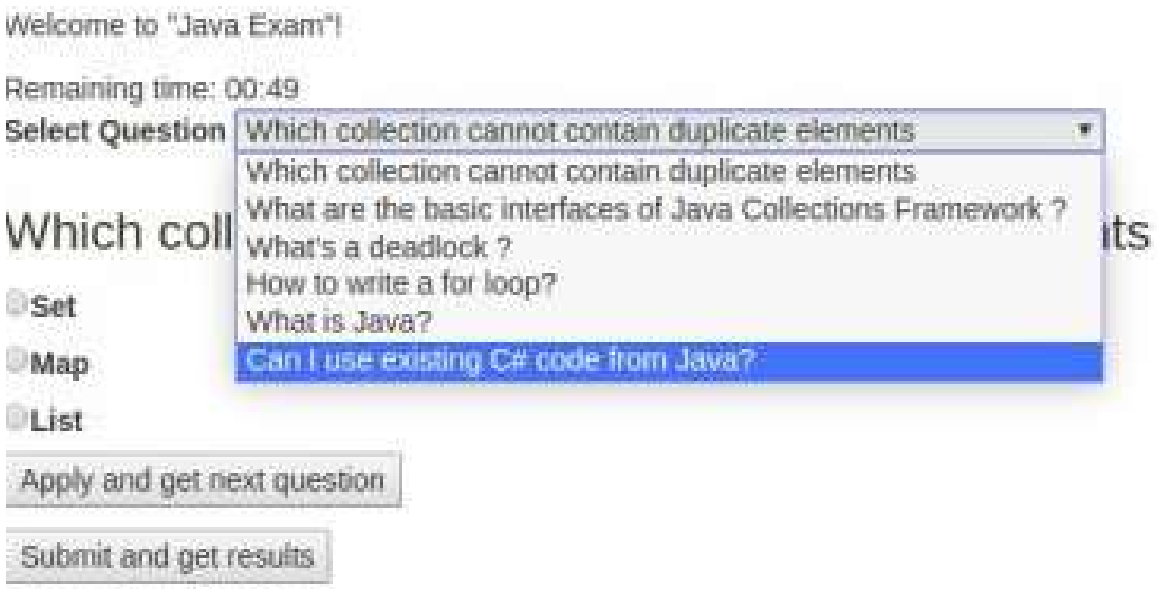


Рис 3.6 Вибір питання

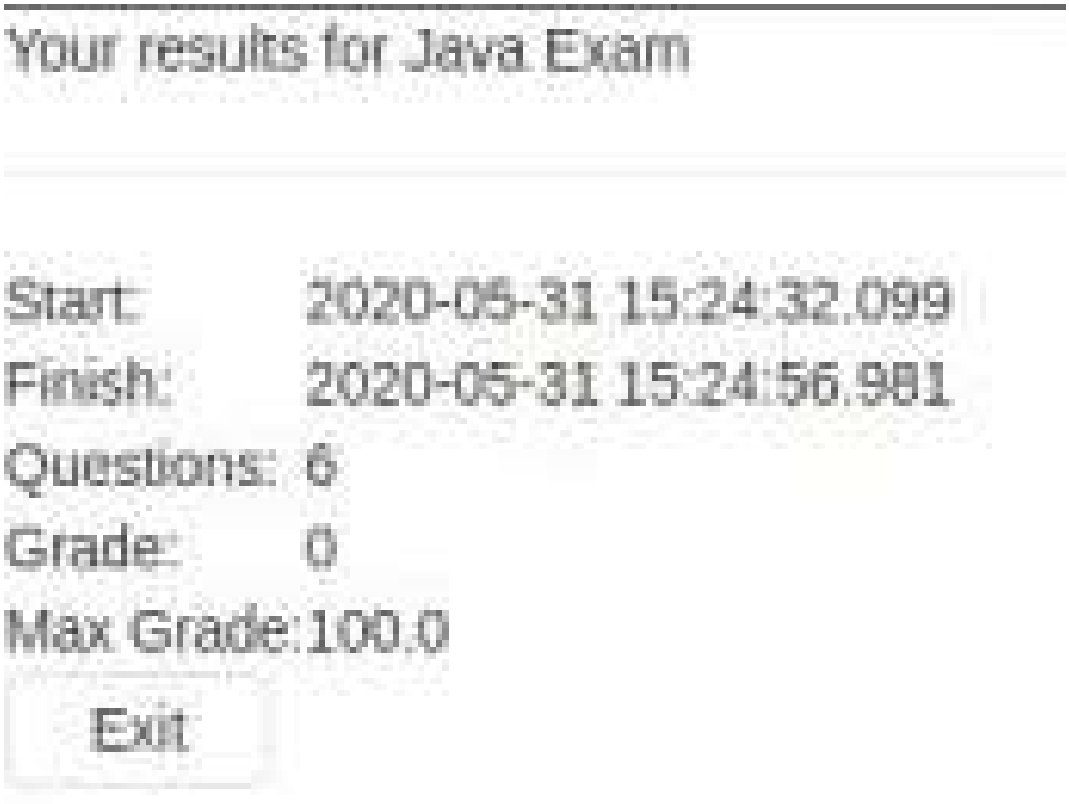


Рис 3.7 Результат тестування

User:	Exam:	Start:	Finish:	Count question:	Answers correct:	Grade:
Ann	Java Exam	2020-05-31 15:50:50.111	2020-05-31 15:51:20.876	6	8	89
Bob	Java Exam	2020-05-31 15:51:33.67	2020-05-31 15:51:49.124	6	2	22
Jon	Java Exam	2020-05-31 15:52:01.239	2020-05-31 15:52:16.717	6	4	44

Рис 3.8 Статистика для перевіряючого

Перевіряючий може побачити кто і коли проходили тест. Побачити оцінку та вид тесту(рис 3.8).

ВИСНОВОК ДО РОЗДІЛУ 3

В даному розділі була розглянута інструкція користувача. Програма має дуже зручний та інтуїтивний інтерфейс, що дозволяє легко пройти тест не шукаючи як і куди потрібно натиснути.

Також були розглянуті усі можливості типи відповідей, як одна правильна так, і декілька. Було розглянуто у якій формі, перевіряючий бачить результати користувачів.

ВИСНОВОК

Результатом бакалаврської роботи є система, підтримки учбового процессу, а, зокрема, тестування. Програма відрізняється від інших аналогів своєю легкістю та зручністю у користуванні. Вона розрахована на користування дітей, які навчаються у закладах середньої освіти, зокрема школах.

Розроблена система має такі можливості як тестування з двома видами питання , одна відповідь або декілька. Всі результати тестування зберігаються у базу даних, тому перевіряючий, зокрема вчитель, бачить результати своїх учнів.

На даному етапі не вдалося реалізувати підтримку хмарних технологій, тобто використовувати систему не локально. Також не вдалося реалізувати кабінети для користувачів, тому що для цього потрібні глибокі знання front end, так як мова Java розрахованна більше на back end системи.

Вона є початковою реалізацією і може бути дороблена по таким аспектам як письмові відповіді на запитання, можливість “деплою” на зовнішні ресурси, привабливіший інтерфейс. Додавання інших підсистем підтримки учбового процессу.

Можна вважати, що система знаходиться на початку своє розробки і, при команді розробників які буду її підтримувати та дороблювати, система може бути використаний у навчальному процесі в школах

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. К. Хорстманн "Java. Бібліотека професіонала. Том 1. Основи " Сп.Б.: "Пітер" 2019.
2. What are Maven goals and phases and what is their difference? [Електронний ресурс]. Режим доступу: https://stackoverrun.com/ru/q/4403362
3. Ant vs Maven vs Gradle: Java Build Tools Comparison [Електронний ресурс]. Режим доступу: https://www.jrebel.com/blog/java-build-tools-comparison
4. What is the history of The Spring Framework? [Електронний ресурс]. Режим доступу: https://www.quora.com/What-is-the-history-of-The-Spring-Framework
5. Google classroom [Електронний ресурс]. Режим доступу: https://classroom.google.com/u/0/h
6. Google forms [Електронний ресурс]. Режим доступу: https://docs.google.com/forms/u/0/
7. Quizlet [Електронний ресурс]. Режим доступу: https://quizlet.com/ru
8. Learning.Apps.org [Електронний ресурс]. Режим доступу: https://learningapps.org/
9. Проєкт «Всеосвіта»Comparison [Електронний ресурс]. Режим доступу: https://vseosvita.ua/

ДОДАТОК 1

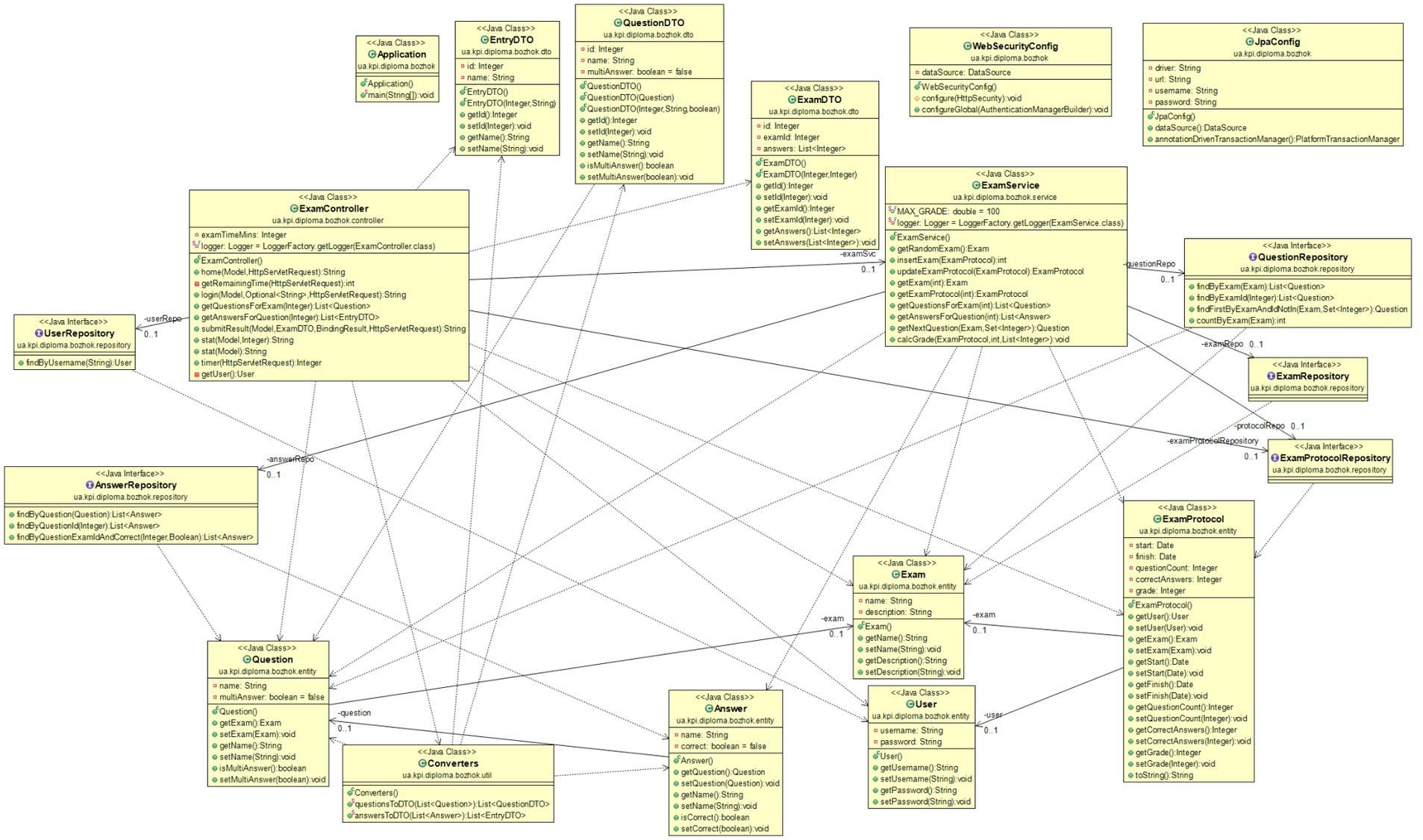
Клієнт-серверна система підтримки учбового процесу

Схема структурна - діаграма класів

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2020 р.



					ІАЛЦ.467100.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Божок Р.Ю.			Схема структурна Діаграма класів	Літ. / Маса / Масштаб
Перевір.		Алещенко О.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П.			Дипломна робота	КПІ ФІОТ кафедра ОТ гр. ІО-63
Затверд.						

ДОДАТОК 2

Клієнт-серверна система підтримки учбового процесу

Схема функціональна - блок-схема алгоритму

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2020 р.

					ІАЛЦ.467100.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	*Схема функціональна Блок-схема алгоритму*	Літ. / Маса / Масштаб
Розроб.		Божок Р.Ю.				
Перевір.		Алещенко О.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П.			*Дипломна робота*	КПІ ФІОТ кафедра ОТ гр. ІО-63
Затверд.						

ДОДАТОК 3

Клієнт-серверна система підтримки учбового процесу

Схема взаємодії

ІАЛЦ.467100.006 ДЗ

Аркушів 1

Київ 2020 р.

					ІАЛЦ.467100.006 Д3	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Божок Р.Ю.			Схема взаємодії	Літ. / Маса / Масштаб
Перевір.		Алещенко О.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П.			Дипломна робота	КПІ ФІОТ кафедра ОТ гр. ІО-63
Затверд.						